Predicting a Song's Popularity Period
Tarun Vallabhaneni
MPCS 53120 – Applied Data Analysis
Final Report
May 19th, 2024

Executive Summary

This project leverages machine learning to analyze and predict song popularity, identifying not only whether a song is a hit but also the specific period in which it would have been popular. By examining detailed audio features, metadata, and lyrical content from extensive music datasets, this project aims to uncover the evolving qualities that influence song popularity over time. The insights gained are particularly valuable for businesses within the music industry, such as radio stations, record labels, and digital music platforms, as they could enhance algorithms for personalized music recommendations. The analysis and models developed in this project offer a deeper understanding of the temporal dynamics of song popularity. Results showed that:

- **Random Forest and XGBoost** were the most effective models, with Random Forest achieving an accuracy of 85.8% and a macro F1-score of 0.68, and XGBoost an accuracy of 85.1% and a macro F1-score of 0.68.
- **Neural Networks** showed potential, especially in predicting pre-2000 hits, with an accuracy of 81.1% and a macro F1-score of 0.68, suggesting a strong capability to understand music characteristics despite class imbalance
- **KNN** had the worst performance, struggling with high-dimensional data and imbalanced classes.

The models were able to predict non-hits and post-2000 hits more accurately, but predicting pre-2000 hits remains a challenge, highlighting the need for more nuanced feature engineering or additional data to capture the complexity of older songs' popularity.

Introduction

Predicting song popularity is a highly debated and complex challenge influenced by various factors, including audio features, lyrical content, cultural trends, and historical context. This project aims to analyze these factors to predict the specific time-period in which a song would have been popular. By examining detailed audio features, metadata, and lyrical content from extensive music datasets, we seek to uncover the evolving qualities that influence song popularity over time. This analysis is valuable for the music industry—radio stations, record labels, and digital music platforms—as it can enhance personalized music recommendation algorithms. Additionally, it allows for a deeper understanding of consumer preferences, enabling greater personalization in both music creation and recommendation. By predicting the optimal popularity period for songs, industry stakeholders can better understand and adapt to changing musical trends and preferences, ensuring their offerings remain relevant and engaging.

Related Work

Study 1: Song Popularity Predictor by Mohamed Nasreldin

This project aimed to develop a model that predicted the likelihood of a song being a hit, defined by its appearance on Billboard's Top 100, with a reported accuracy of over 68% (Nasreldin, 2018).

Data Used and Preprocessing

- **Million Song Dataset:** This study employed the Million Song Dataset provided by Columbia University, including 41 features categorized into audio analysis (e.g., tempo, duration), artist-related features (e.g., artist familiarity), and song-related features (e.g., releases, title, year).
- **Dataset Transition:** Due to the difficulty in obtaining a complete dataset from the Million Song Dataset (MSD) and the insufficiency of a random subset for generating proper insights, I transitioned to using the Spotify Dataset 1921-2020, 600k+ Tracks from Kaggle.

- **Data Cleaning:** Nasreldin's project dropped many fields due to outdated and missing data and filled some missing values with the mean. My dataset had different fields with fewer outdated and missing data values, so I did not have to employ many of these techniques. However, I cleared out empty data values and performed minimal data cleaning to ensure data integrity.

Data Collection Methods

- **Billboard Data Scraping:** Their study scraped Billboard Top 100 data to classify songs as hits, providing an objective benchmark for song popularity. This was crucial for my project as it taught me how to scrape for the Billboard dataset. I ended up using their scraped data (publicly available on GitHub) to avoid re-scraping, which allowed me to create a robust popularity metric.

Models Considered and Used

The study utilized various models, including XGBoost (XGB), Logistic Regression, Random Forest, K-Nearest Neighbors (KNN), Decision Tree, and Support Vector Machines (SVM).

Metrics for Performance Evaluation

- **Classification Metrics:** Precision, recall, F1 score, and AUC (Area Under the Curve). While they used AUC as their primary metric, suitable for binary classification, I used F1 scores as my primary metric to better handle the multi-class classification problem in my project.

Evaluation

- **Feature Analysis and Model Selection:** The extensive feature analysis conducted in the project underscored the importance of robust feature evaluation. Their findings revealed that non-linear relationships may dominate, guiding me to prioritize non-linear models like XGBoost, Random Forest, and KNN in my study. XGBoost provided the best predictions on the training model used in the study, with an AUC score of 0.68 (Random predictions would yield a 0.5 AUC score). Although this project varied from mine in that mine was a multi-class classification problem, their findings significantly influenced my model choice.
- **Hyperparameter Tuning:** The study performed grid search for hyperparameter tuning on XGBoost, adjusting parameters like n_estimators and learning_rate to improve model performance. This approach increased the AUC score from 0.632 to 0.68. Their strategy for hypertuning directly influenced my own approach to model optimization.

Results

- **Feature Analysis:** The study identified the most influential features in their dataset, including artist familiarity, loudness, year, and genre tags. Acoustic features were found to be less predictive than metadata features due to their variability within a single song.
- **Model Performance:** XGBoost provided the best predictions on the training model, with an AUC score of 0.68.

Challenges and Considerations

- **Addressing Data Skewness:** The previous study noted a challenge in predicting hit songs due to data skewness—only 1,200 of their songs were hits. This influenced their model's ability to predict non-hits more accurately than hits. This was like some issues I faced. This study made me aware of data skewness issues and provided a good verification for my own model.

Implications for My Project

- **Approaches/Techniques:** The study highlighted the importance of robust feature evaluation and non-linear models like XGBoost. Their approach to handling missing data, scraping Billboard data, and hypertuning directly influenced my methodology. I integrated their dataset for Billboard data, utilized non-linear models, and adopted their hypertuning strategy.

- **Benchmarks:** This study provided benchmarks for model performance, such as AUC scores, which I used to evaluate the success of my own models.

Study 2: Predicting Song Popularity by James Pham, Edric Kyauk, and Edwin Park

This Stanford University study aimed to predict song popularity using the Million Song Dataset, evaluating various machine learning models for effectiveness. Their research underscored the complexity of predicting song hits based on acoustic features and metadata (Pham, 2015)

Data Used

- **Million Song Dataset:** An exhaustive collection of audio features and metadata for approximately one million songs dating to 2011. The audio features included attributes about the music track itself, such as duration, key, and year. The metadata used more abstract features, such as danceability, energy, and song hotttnesss, generated from The Echo Nest, a music intelligence platform.
- **Data Selection Strategy**: The Stanford team extracted 10,000 tracks from the Million Song Dataset and removed all tracks that were missing any of the features they were considering, leaving them with 2,717 tracks. They divided these tracks so that 90% of the tracks were used for training and 10% for testing. In contrast, I used a more meticulous selection strategy from the Spotify dataset, ensuring a balanced and chronologically diverse dataset that represented both hits and non-hits effectively. This strategic selection was essential to analyze trends over different periods accurately.

Data Preprocessing

- **Feature Extraction:** The Stanford team included baseline features such as the mean and variance of acoustic features across song segments. They also added additional features to capture polynomial relationships and interaction terms. My dataset had different fields with fewer outdated and missing data values. Therefore, I did not have to employ many of the data cleaning techniques used in their paper.
- **Feature Engineering**: Their use of L1 regularization and forward feature selection to enhance model generalization by reducing feature dimensionality was particularly insightful. Additionally, their methodology for applying transformations to predictors, processing array-type features, and using bag of words to capture string features were all crucial strategies

Models Considered and Used

The study utilized a variety of models including Logistic Regression with L1 regularization, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA) with class-specific covariance matrices, Support Vector Machines (SVM), and Multilayer Perceptron (MLP) neural networks. Additionally, multiple linear regression and Lasso regression were used to predict song popularity values.

Metrics for Performance Evaluation

- **Classification Metrics:** Precision, recall, F1 score, and AUC (Area Under the Curve).
- **Regression Metrics:** Mean Squared Error (MSE) was used to evaluate how well the model predicted popularity, with the average error approximating the standard error of the predictions.

Evaluation

- **Feature Selection:** Forward stepwise selection chose a model of 45 features, while backward stepwise selection chose a model of 81 features. Significant features included artist familiarity, loudness, year, and genre tags such as "alternative," "guitar," and "jazz."

- **Classification Results:** The SVM with a Gaussian kernel had the highest F1 score, indicating that non-linear relationships between features and popularity were important. However, F1 scores for all models ranged between 0.5 and 0.6, and accuracies ranged between 0.75 and 0.80, suggesting that the data was not linearly separable.
- **Regression Results**: Lasso regression achieved the smallest test error among the models by using cross-validation to choose the optimal value of λ. The selected model with Lasso regression had a significantly lower MSE compared to the full model with all features.

Results

- **Feature Analysis:** The study identified the most influential features in their dataset, including artist familiarity, loudness, year, and genre tags. Acoustic features were found to be less predictive than metadata features due to their variability within a single song.
- **Model Performance:** The SVM with a Gaussian kernel performed best in classification tasks, while Lasso regression performed best in regression tasks.

Implications for My Project

- **Approaches/Techniques:** The study highlighted the importance of feature selection and the effectiveness of non-linear models. It also demonstrated the value of using additional features and interaction terms to capture complex relationships. The insights from this study informed the selection of models and feature engineering techniques in my project.
- **Benchmarks:** This study provided benchmarks for model performance, such as F1 scores and AUC values, which I used to evaluate the success of my own models.

Study 3: Predicting Song Popularity Based on Lyrics by Miguel Cozar

This project aimed to predict song popularity using lyrical content, evaluating various machine learning models to determine the effectiveness of lyrics as a predictor for song hits (Cozar, 2022).

Data Used

- **Data Source**: The study used lyrics from genius.com, processed using the genius.com API. Non-English songs were translated to English using the Google Translator API.
- **Text Processing**: Lyrics were transformed into vectors using BERT sentence transformers from HuggingFace, resulting in vectors of 768 dimensions.

Data Preprocessing

- **Dimensionality Reduction**: Principal Component Analysis (PCA) was used to reduce the dimensions of the dataset to 100, simplifying the training process without losing significant information. Similarly, in my project, I used PCA to reduce the dimensions of genre embeddings derived from BERT sentence transformers to ensure the training process was efficient and effective.

Models Considered and Used

The study utilized a deep neural network model built with Keras, containing layers for convolution, max pooling, and dense layers. The network was trained to minimize mean squared error (MSE) using the Adam optimizer.

Metrics for Performance Evaluation

- **Regression Metrics**: MSE was used to evaluate model performance. The MSE obtained in the test phase was 0.0027, indicating close predictions to actual values.

Evaluation

- **Dimensionality Reduction and Visualization**: PCA was used to visualize the dataset in 3D, showing a clear relationship between lyrics and song popularity. Similarly, I used PCA to reduce and visualize the high-dimensional genre data, which had over 4500 unique labels.
- **Model Training and Testing**: The neural network was trained on 4000 samples and tested on 1000 samples. The training results showed that the model did not overfit and provided accurate predictions during the test phase.

Results

- **Model Performance**: The neural network achieved an impressive MSE of 0.0027 in the test phase, demonstrating the effectiveness of using lyrics for predicting song popularity.
- **Error Analysis**: The model's error distribution was centered around zero, forming a Gaussian curve, indicating no significant tendency to overestimate or underestimate popularity.

Implications

- **Approaches/Techniques**: This study's methodology for processing text data and using BERT sentence transformers inspired my approach to incorporate genre information into my model. I used PCA for dimensionality reduction of the BERT embeddings, similar to their approach. While I could not use MSE as my project is a classification problem, I learned to build neural networks using Keras and to use Sentence Transformers from this study. This informed my techniques for handling large text data and creating embeddings for complex features and my model choice.
- **Adaptation of Techniques**: Although I initially aimed to include lyrical content, the lack of sufficient data led me to focus on genre tags. I applied similar techniques to analyze genre similarities, using the large number of unique genre labels in my dataset, which had over 4500 unique labels.

Methodology

Effort Description

The project involved several key activities:

- **Data Collection and Preprocessing:** Approximately 30% of the effort was dedicated to gathering and cleaning the data from various sources. This included integrating datasets, handling missing values, and normalizing data formats.
- **Feature Engineering:** About 20% of the time was spent on feature engineering, including generating genre embeddings and reducing dimensionality using PCA, which I learnt from class.
- **Model Training:** 50% of the effort went into training the models, performing cross-validation, and tuning hyperparameters to optimize performance. I had to independently learn to use Keras.
- **Evaluation and Analysis:** The remaining 20% was focused on evaluating model performance, tuning and analyzing results, and interpreting the findings.

Dataset Used

- **Spotify Dataset 1921-2020, 600k+ Tracks**: This dataset from Kaggle includes over 600K+ songs dating back to the 1920s. It contains audio features, metadata, and genre information, all scraped from the Spotify API. One CSV file contained all the songs, another CSV included the artists and genre tags, and a JSON dictionary was used to merge the datasets. The dataset can be found here.
- **Billboard Data**: Scraped using Python and BeautifulSoup, this data includes historical popularity metrics for songs that appeared on the Billboard Top 100. I used publicly available data from a GitHub repository, avoiding the need to re-scrape. The dataset can be found here.

- **Dataset Change**: Initially, I intended to use the Million Song Dataset, but due to the difficulty in obtaining the data, I changed my approach. The Million Song Dataset only provided random subsets of 10,000 songs, with each song as an individual h5 file, making the parsing and data collection process very challenging and time-consuming. I required a larger and more comprehensive dataset, which led me to use the Spotify Dataset from Kaggle. This dataset offered a more substantial volume of data, ensuring a well-represented dataset for my analysis.

Software Used

- **Python Libraries**:
    - **Pandas**: For data manipulation.
    - **NumPy**: For numerical computations.
    - **Scikit-learn**: For machine learning models and preprocessing.
    - **imbalanced-learn**: For applying SMOTE.
    - **Keras**: For building neural networks.
    - **HuggingFace**: For BERT sentence transformers used to convert genre text into embeddings.

Data Preprocessing and Feature Engineering

- **Loading and Cleaning Data**

The Spotify dataset was initially loaded, and duplicate songs were removed based on the name and artist. The artist dataset was then loaded, and artist IDs were normalized. The artist, genre tags and song information were combined. Missing data were handled by dropping rows with missing values to ensure data integrity. All categorical data was also converted before being used in the models.

- **Exploding Songs with Multiple Artists**

Songs with multiple artists were exploded to better delineate different artists, resulting in a dataset of **570,548 songs**. This approach allowed for a more detailed analysis of the influence of individual artists.

- **Billboard Data Integration**

To create a robust popularity metric, Billboard data were merged, song titles were normalized, and indicators for Billboard hits were created. Initially, 51,696 songs were identified as Billboard hits, representing a small percentage of the dataset. To better indicate popularity, I calculated the quartiles of the popularity score provided by Spotify:

- **First quartile** (25th percentile): 10.0, **Median** (50th percentile): 25.0, **Upper quartile** (75th percentile): 40.0

Using a cutoff score of 50 for popular songs, an additional 64,089 songs were classified as popular. The final dataset had **70,181** Post-2000s Hit songs, **34,498** Pre-2000s Hit songs, and **465,869** Non-Hit songs, leading to a benchmark of 81.65% accuracy if the model guessed everything as a non-hit.

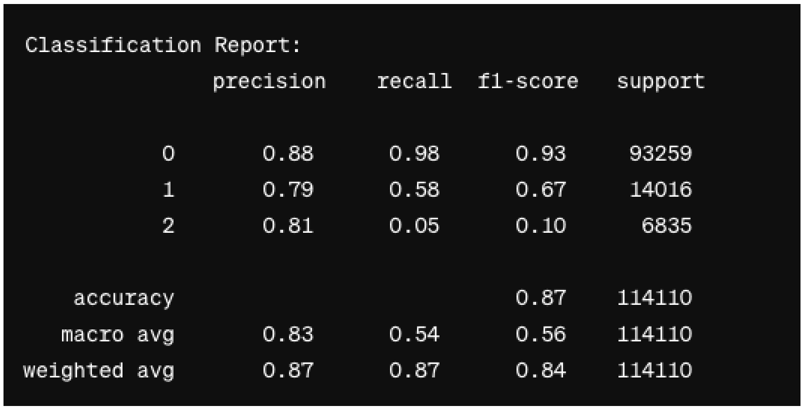
- **Feature Engineering**

Genre tags were included as an important feature for classification. The genre lists were converted into strings, and the data were saved after cleaning and processing. Using Sentence Transformers, 768-dimensional embeddings for genre tags were generated, and PCA was applied to reduce the dimensionality to 50. Release dates were converted to decade of release, which was further categorized into Pre-2000s Hits, Post-2000s Hits and Non-Hits in the **hit_category_encoded** columns of the data.

Initial Model Attempts

- **Random Forest Model without Genres**

An initial model was trained using Random Forest without genre classification, yielding poor results.



```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.98      0.93     93259
           1       0.79      0.58      0.67     14016
           2       0.81      0.05      0.10      6835

    accuracy                           0.87    114110
   macro avg       0.83      0.54      0.56    114110
weighted avg       0.87      0.87      0.84    114110
```

Figure 1: Random Forest Classification Report (without genres)

- **Interpretation of Results**

The model struggled to classify pre-2000 hits, achieving an F1 score of 0.10. This indicated that the audio features provided by Spotify did not fully encompass all the characteristics needed for accurate classification. This led to the hypothesis that incorporating genre information could improve model performance, as genre trends vary over time and provide valuable context for classification.

- **Initial One-Hot Encoding Approach**

Initially, songs were classified by their popularity in specific decades using one-hot encoding for each decade. This approach resulted in F1 scores of 0 for all categories below the 1980s, indicating high similarities within categories and insufficient data to delineate them properly. This failure led to two changes in approach: adding genre information and reducing the classification to three categories.

- **Changing Classification Approach**

The classification approach was revised to categorize songs into Pre-2000s hits, Post-2000s hits, and Non-hits. This change aimed to better capture temporal trends in music popularity. However, genre tags posed a challenge due to their complexity (over 4,672 unique labels) and the presence of phrases or emotions rather than pure genres.

- **Preliminary Analysis of Genres**

Preliminary analysis revealed over 4,672 unique genres, with more than 500 genres having over 1,000 songs each. This complexity made a dictionary approach infeasible, leading to the use of Natural Language Processing (NLP) techniques.

- **Using Sentence Transformers**

Given the limitations of Word2Vec for handling multiple tags and phrases, Sentence Transformers were used to generate 768-dimensional embeddings for genre tags. This allowed for the effective incorporation of genre information into the model. This approach was inspired by Study 3, which effectively incorporated textual data into neural networks using similar techniques.

- **Dimensionality Reduction with PCA**

PCA was applied to reduce the dimensionality of the genre embeddings, settling on 50 features that captured 75% of the explained variance. Initial tests with 100 PCA features indicated potential overfitting, as accuracy dropped slightly on the Random Forest, confirming that 50 features were sufficient.

- **Random Forest Model with Genres**

Random Forest was selected for initial modeling due to its reported effectiveness in prior studies and its ability to handle unscaled features. The inclusion of genre embeddings significantly improved the model's performance, as will be demonstrated in the results section.

Feature Analysis

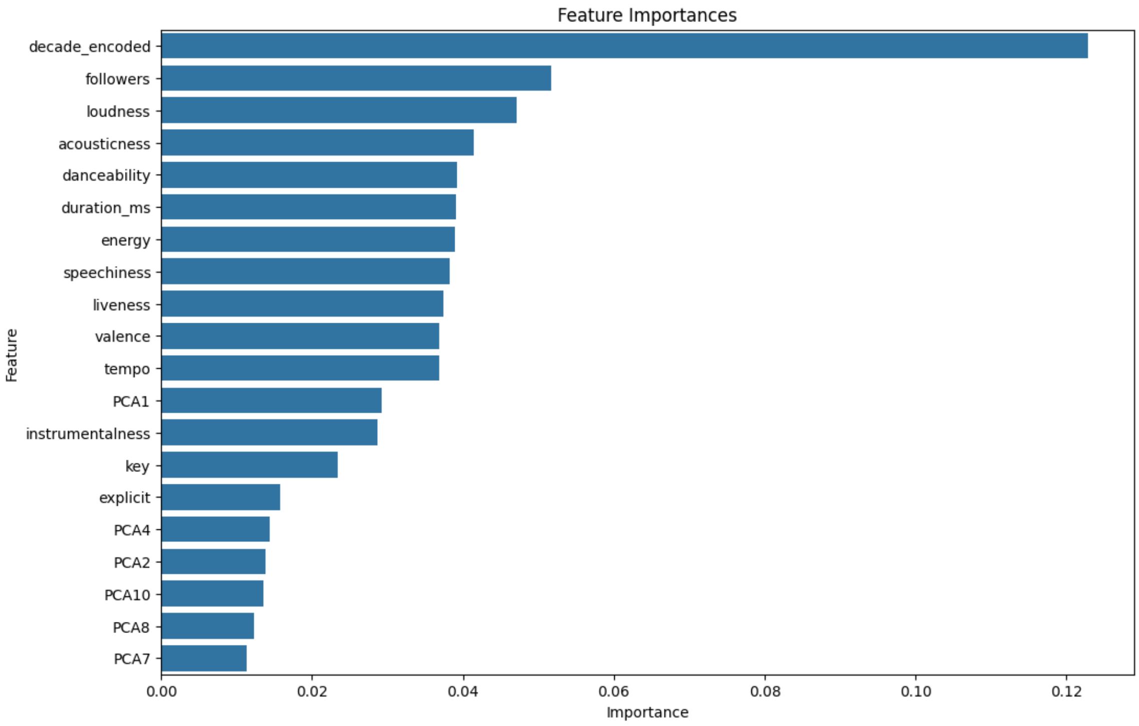


Figure 2: Top 20 Important Features (excluding genre embeddings)

The decade in which a song was released heavily impacts its popularity. Musical trends and preferences vary greatly over time, making the temporal context crucial for accurate classification. I kept the decade in the analysis to evaluate whether audio features alone could be more significant indicators of popularity than the release date, but it appears the decade is still a dominant factor. As suspected, the number of followers an artist has is a major predictor of song popularity. This shows that an artist's popularity significantly influences song popularity, regardless of the music itself. Due to the PCA transformation, it's challenging to pinpoint the specific influence of individual genres. However, the inclusion of genre information contributes to the overall model performance, capturing complex patterns that affect song popularity, as shown below.

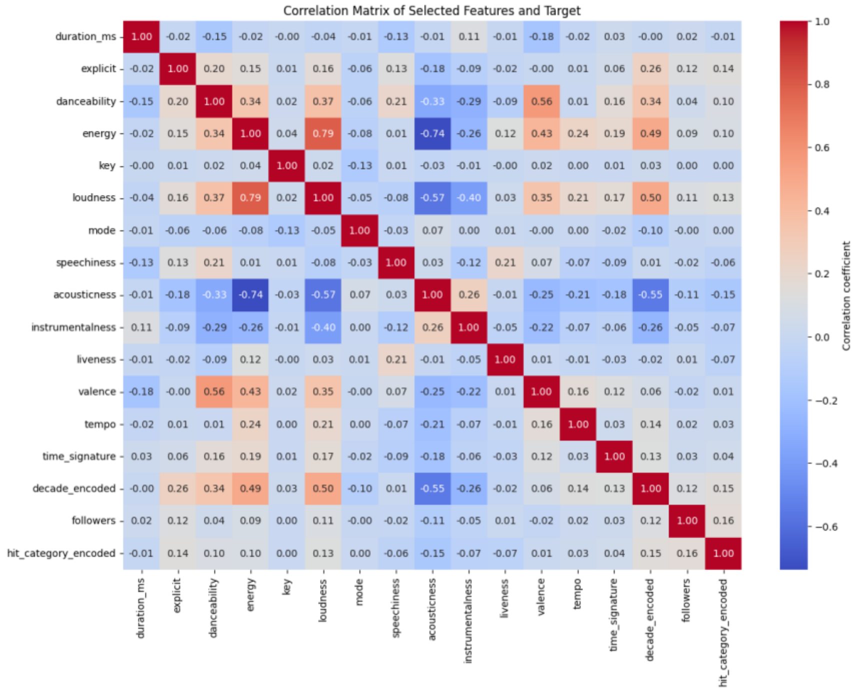


Figure 3: Correlation Matrix (including genre embeddings)

In addition, the correlation matrix analysis revealed significant correlations between some features:

- **Acousticness and Energy**: These features have a -0.74 correlation, indicating that as one increases, the other tends to decrease.
- **Energy and Loudness**: These features have a 0.79 correlation, indicating a strong positive relationship.

A threshold of 0.70 was used to potentially eliminate correlated features if needed, ensuring that the model's performance is not adversely affected by multicollinearity.

Models and Results

Train-Test Splits

All models were split into a test and validation set. Specifically, 80% of the data was allocated for the training set and 20% for the test set. 20% of the training set was further split to form the validation set. The validation set was used to hypertune the models, and the final evaluation was conducted on the test set.

SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE was used to address class imbalances in the dataset, where less than 20% of songs were classified as hits, and within this subset, there was further division into pre-2000s and post-2000s hits. By generating synthetic samples for the minority classes, SMOTE helped create a more balanced training dataset. This technique ensured the classifier received adequate examples from all classes, improving the model's ability to learn and generalize, leading to better predictions for minority class samples and enhancing the robustness of the classification results (K, 2021), despite leading to lower accuracy figures.

Random Forest

The Random Forest Classifier is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputting the classification)of the individual trees. It is well-suited for handling datasets with high-dimensional feature spaces and can manage both categorical and numerical features without the need for extensive preprocessing, making it a suitable choice to quickly test the impact of including genre embeddings versus excluding them . Random Forests are robust against overfitting, particularly when working with large datasets, and they provide a measure of feature importance, which helps in understanding the influence of various features on the classification outcomes (*What is Random Forest?*, 2021).

Initial Results Without Genre Embeddings:

- **Accuracy:** 0.8723
- **Classification Report:**
  - **Class 0 (Non-hits):** Precision: 0.88, Recall: 0.98, F1-score: 0.93
  - **Class 1 (Post-2000 hits):** Precision: 0.79, Recall: 0.58, F1-score: 0.67
  - **Class 2 (Pre-2000 hits):** Precision: 0.81, Recall: 0.05, F1-score: 0.10
  - **Macro Average:** Precision: 0.83, Recall: 0.54, F1-score: 0.56
  - **Weighted Average:** Precision: 0.87, Recall: 0.87, F1-score: 0.84

Interpretation:

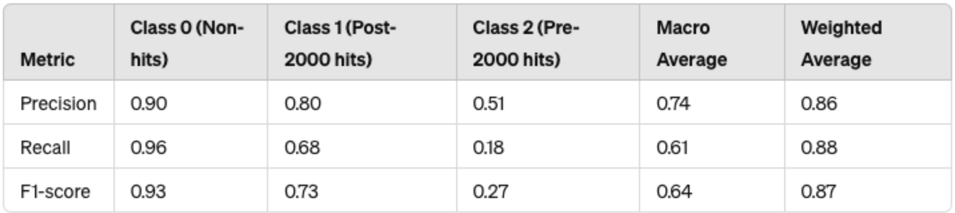
The initial Random Forest model, trained without genre embeddings, yielded a high overall accuracy of 87.23%, surpassing the benchmark expected random accuracy of 81.65%. However, the model's performance varied significantly across different classes.

- **Non-hits (Class 0):** Based on the classification report, the model was very effective at identifying non-hits, likely because this class had the most data points, leading to better prediction.
- **Post-2000 hits (Class 1):** The lower recall for this class suggests that the model struggled to identify all post-2000 hits, likely due to fewer data points and potential overlap with the non-hits.
- **Pre-2000 hits (Class 2):** The model's performance was poor for this class, indicating that the model was not effective at identifying pre-2000 hits. The low recall suggests that many pre-2000 hits were misclassified, potentially due to insufficient distinguishing features in the provided audio and metadata without genre information

Results:

Random Forest with PCA (No SMOTE):

- **Test Accuracy:** 0.879
- **Classification Report:**

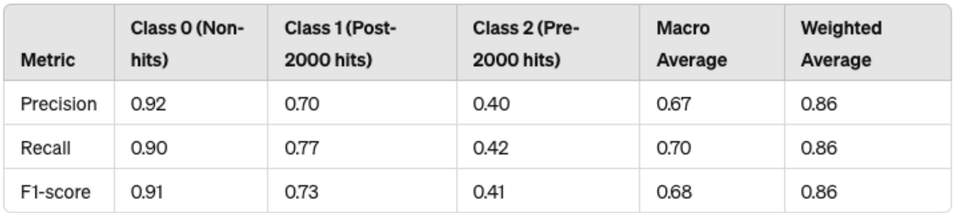


Metric	Class 0 (Non-hits)	Class 1 (Post-2000 hits)	Class 2 (Pre-2000 hits)	Macro Average	Weighted Average
Precision	0.90	0.80	0.51	0.74	0.86
Recall	0.96	0.68	0.18	0.61	0.88
F1-score	0.93	0.73	0.27	0.64	0.87

Figure 4: Random Forest Classification Report (No SMOTE)

Random Forest with PCA and SMOTE:

- **Test Accuracy:** 0.858
- **Classification Report:**



Metric	Class 0 (Non-hits)	Class 1 (Post-2000 hits)	Class 2 (Pre-2000 hits)	Macro Average	Weighted Average
Precision	0.92	0.70	0.40	0.67	0.86
Recall	0.90	0.77	0.42	0.70	0.86
F1-score	0.91	0.73	0.41	0.68	0.86

Figure 5: Random Forest Classification Report (SMOTE)

- **Confusion Matrix:**

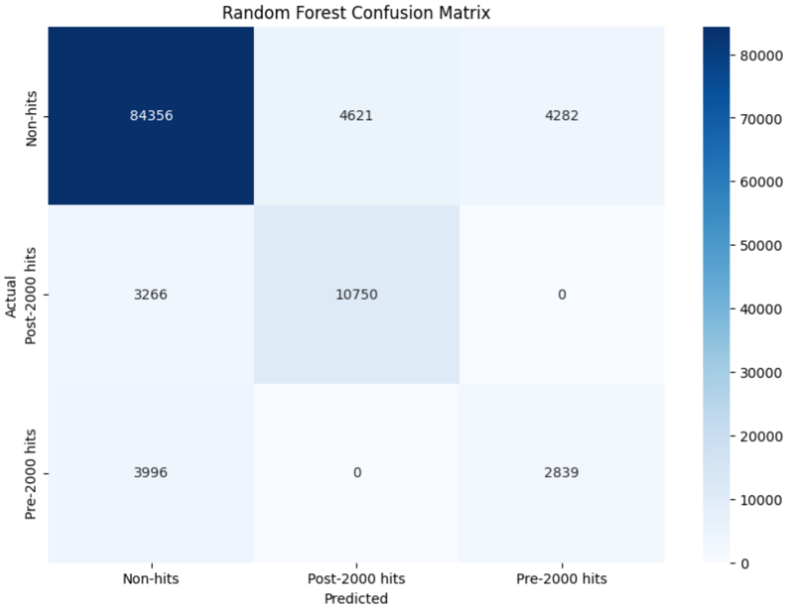


Figure 6: Random Forest Confusion Matrix

Model Insights:

1. **Improvement with SMOTE:** Applying SMOTE helps to balance the dataset, improving recall and F1-scores for minority classes, especially Class 2 (Pre-2000 hits), though it reduces accuracy.
2. **Challenges with Class 2:** Both Random Forest models, with and without SMOTE, struggled with accurately predicting Class 2 (Pre-2000 hits). The low precision and recall for this class indicate that the model finds it challenging to distinguish older hits from non-hits and post-2000 hits. This difficulty is likely due to the smaller sample size of pre-2000 hits in the dataset and the complex, evolving nature of musical trends over time. The F1-score for Class 2 improved with SMOTE, but it remains the lowest among the classes, suggesting the need for more refined feature engineering or additional data.
3. **Confusion Matrix:** The model had good performance in identifying non-hits and post-2000 hits, correctly classifying 84,356 non-hits and 10,750 post-2000 hits. However, it struggles with pre-2000 hits, correctly identifying only 2,839 and misclassifying 3,996 as non-hits. The model shows a clear distinction between post-2000 and pre-2000 hits, with no post-2000 hits

misclassified as pre-2000 hits. This indicates that while the model effectively captures temporal differences, it faces challenges in accurately identifying older hits.

4. **Genre Embeddings Impact:** Including genre embeddings and applying PCA enhances the model's ability to capture complex patterns, significantly improving performance across most classes. This improvement is reflected in the overall classification metrics, with higher macro and weighted averages when genre embeddings are included.
5. **Hypertuning:** Changing the max depth and the number of estimators did not significantly alter the classification matrix for Random Forest. This indicates that the model's performance is relatively stable with respect to these hyperparameters

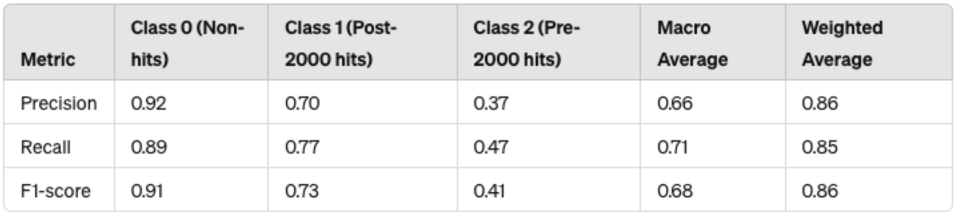
XGBoost

XGBoost is a machine learning gradient boosting algorithm that uses a regularization technique in it. In simple words, it is a regularized form of the existing gradient-boosting algorithm. Due to this, XGBoost performs better than a normal gradient boosting algorithm and that is why it is much faster than that also. Its efficiency and effectiveness in handling structured data make it an ideal choice for this project (*XGBoost*, 2023). Previous studies, such as Study 1 (Nasreldin, 2018), demonstrated its capability in predicting song popularity.

For XGBoost, after experimenting with various hyperparameters, the final model was configured with the following parameters: **objective='multi:softprob', eval_metric="mlogloss", use_label_encoder=False, n_estimators=500, learning_rate=0.3, random_state=42**.

Results with XGBoost (PCA, SMOTE):

- **Test Accuracy:** 0.851
- **Classification Report:**



Metric	Class 0 (Non-hits)	Class 1 (Post-2000 hits)	Class 2 (Pre-2000 hits)	Macro Average	Weighted Average
Precision	0.92	0.70	0.37	0.66	0.86
Recall	0.89	0.77	0.47	0.71	0.85
F1-score	0.91	0.73	0.41	0.68	0.86

Figure 7: XGBoost Classification Report

- **Confusion Matrix:**

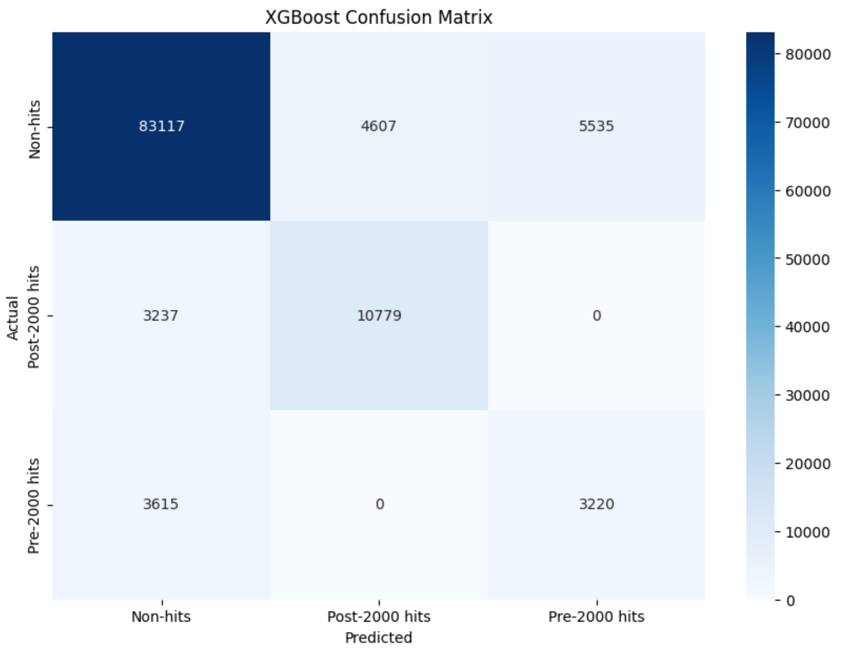


Figure 8: XGBoost Confusion Matrix

Interpretation:

- The accuracy was slightly lower at 0.8580 than without SMOTE (0.8793 but struggled with recall for Class 1 and 2). Post applying SMOTE, Class 0 had high precision and recall, maintaining a strong F1-score. Class 1 (Post 2000s hits) saw improved recall due to SMOTE, which balanced the class distribution, leading to a higher F1-score. Class 2's performance also improved, with better recall and F1-score.
- SMOTE improved the classification of hits, leading to better recall and F1-scores for hits (both Post-2000 and Pre-2000 categories). Although the overall accuracy was slightly lower than without SMOTE (0.851 vs. 0.881), the macro F1-score improved (0.64 vs 0.68), indicating more balanced predictions. The high accuracy without SMOTE was likely due to the model's bias towards non-hits, as indicated by the high precision and recall for non-hits (0.90 precision and 0.96 recall), but poorer macro F1-scores.
- According to the confusion matrix: The model correctly identifies 83,117 non-hits, 10,779 post-2000 hits, and 3,220 pre-2000 hits. However, it misclassifies a notable number of non-hits as hits and hits as non-hits, particularly pre-2000 hits. This indicates that while the model is effective at identifying non-hits and post-2000 hits, it has difficulty distinguishing older hits, highlighting the challenge of classifying pre-2000 hits accurately.

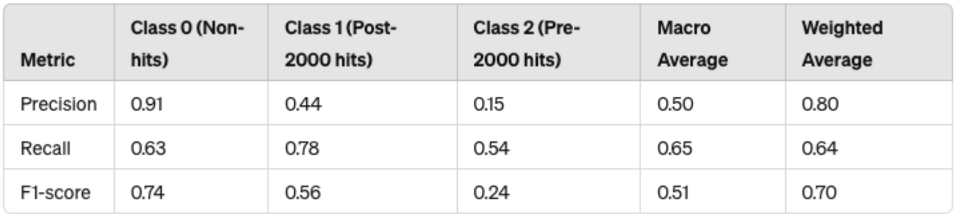
K-Nearest Neighbors (KNN)

The K-Nearest Neighbors algorithm classifies a data point based on how its neighbors are classified. For each point in the test set, the algorithm finds the 'k' closest data points in the training set and assigns the most common class among those neighbors to the test point. KNN is useful for its simplicity and effectiveness in classification tasks where the decision boundary is complex and non-linear (Lavasani, 2024).

Scaling the Data: KNN relies on distance calculations, making it sensitive to the scale of the data. Therefore, I scaled the features to ensure that each feature contributes equally to the distance measurements.
Feature Reduction: During collinearity analysis, two features ('loudness' and 'acousticness') with a correlation greater than ±0.7 were removed to reduce redundancy and improve model performance.
Hypertuning: After testing various numbers of neighbors, it was determined that using 3 neighbors yielded the best results. Increasing the number of neighbors beyond this point decreased accuracy.

Results with KNN (n=3, PCA, SMOTE, Reduced Features):

- **Test Accuracy:** 0.644
- **Classification Report:**

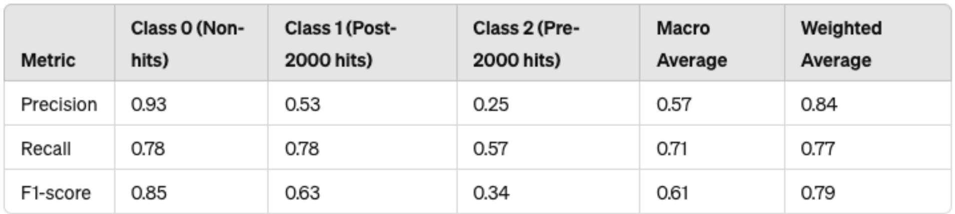


Metric	Class 0 (Non-hits)	Class 1 (Post-2000 hits)	Class 2 (Pre-2000 hits)	Macro Average	Weighted Average
Precision	0.91	0.44	0.15	0.50	0.80
Recall	0.63	0.78	0.54	0.65	0.64
F1-score	0.74	0.56	0.24	0.51	0.70

Figure 9: KNN Classification Report (Reduced Features)

Results with KNN (n=3, PCA, SMOTE, All Features):

- **Test Accuracy:** 0.766
- **Classification Report:**



Metric	Class 0 (Non-hits)	Class 1 (Post-2000 hits)	Class 2 (Pre-2000 hits)	Macro Average	Weighted Average
Precision	0.93	0.53	0.25	0.57	0.84
Recall	0.78	0.78	0.57	0.71	0.77
F1-score	0.85	0.63	0.34	0.61	0.79

Figure 10: KNN Classification Report (All Features)

- **Confusion Matrix:**

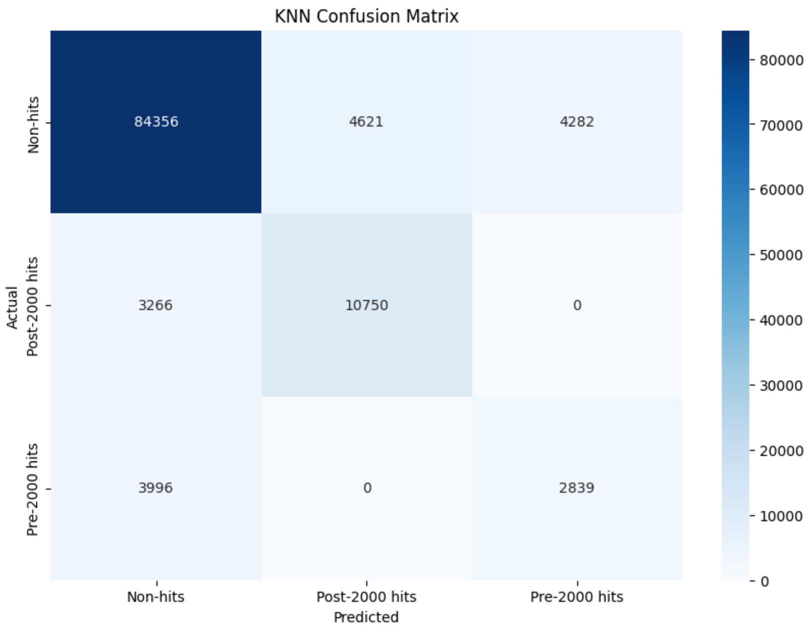


Figure 11: KNN Confusion Matrix (All features)

Interpretation:

KNN with PCA, SMOTE, and reduced features yielded a test accuracy of 64.4%, which is lower than previous models. The precision and recall for Class 0 (Non-hits) were relatively high, but the performance significantly dropped for Class 1 (Post-2000 hits) and Class 2 (Pre-2000 hits). This model achieved a macro F1-score of 0.51, indicating challenges in distinguishing between the hits and non-hits. KNN struggled with the high-dimensional and imbalanced nature of the dataset, even after feature reduction.

The removal of the 'loudness' and 'acousticness' features, identified during collinearity analysis, did not significantly improve the model's performance, further emphasizing the complexity of accurately predicting song popularity using this approach. Interestingly, leaving all features in the model improved performance, achieving a test accuracy of 76.6% and a macro F1-score of 0.61, and better cassified Class 1 and Class 2 hits compared to the reduced feature set. This indicates that despite potential collinearity, the excluded features provided valuable information for classifying hits and non-hits. The confusion matrix shows that while the model effectively identified non-hits and post-2000 hits, it struggled significantly with pre-2000 hits, misclassifying a substantial number as non-hits. Overall, the KNN model had difficulty accurately classifying pre-2000 hits and managing the complexities of the dataset.

Neural Networks

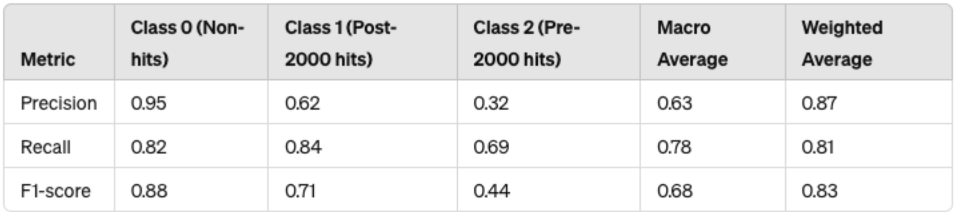
Neural networks are computational models that consist of interconnected layers of nodes that process input data, making them highly effective for capturing complex, non-linear relationships in high-dimensional datasets (*What is a neural network?,* 2021). Neural networks are capable of learning intricate patterns from large datasets. Given the high-dimensional nature of genre embeddings and audio features, neural networks were chosen to effectively capture the subtle relationships and interactions.

Neural Network Architecture:

- **Input Layer:** 128 neurons, ReLU activation
- **Hidden Layer 1:** 64 neurons, ReLU activation
- **Hidden Layer 2:** 32 neurons, ReLU activation
- **Dropout Layers:** Three dropout layers with a dropout rate of 0.3 to prevent overfitting
- **Output Layer:** Softmax activation to classify the three categories (Non-hits, Post-2000 hits, Pre-2000 hits)
- **Optimizer:** Adam optimizer with a learning rate of 0.001
- **Loss Function:** Categorical cross-entropy
- **Training:** 30 epochs, batch size of 64

Results with Neural Network (PCA, SMOTE):

- **Test Accuracy:** 0.811
- **Classification Report:**



Metric	Class 0 (Non-hits)	Class 1 (Post-2000 hits)	Class 2 (Pre-2000 hits)	Macro Average	Weighted Average
Precision	0.95	0.62	0.32	0.63	0.87
Recall	0.82	0.84	0.69	0.78	0.81
F1-score	0.88	0.71	0.44	0.68	0.83

Figure 12: Neural Network Classification Report

- **Confusion Matrix:**

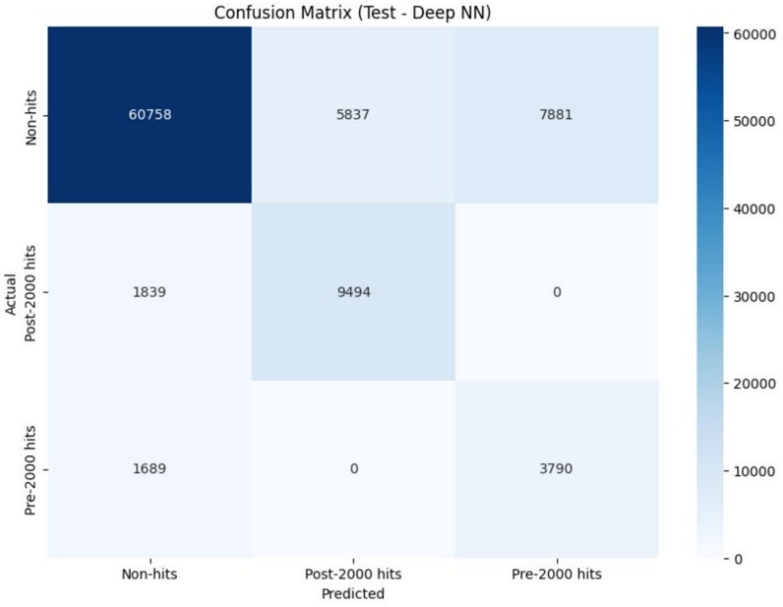


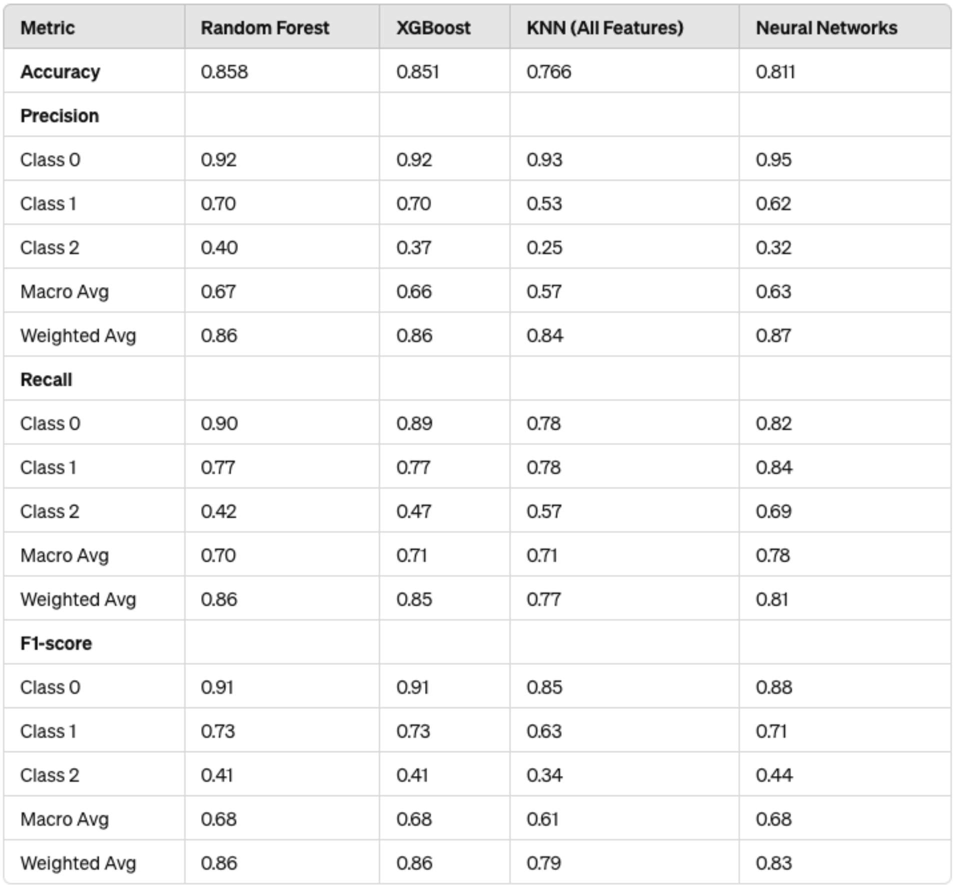
Figure 13: Neural Network Confusion Matrix

Interpretation:

The neural network achieved a test accuracy of 81.81% and a macro F1 score of 0.68. The model performed well in classifying non-hits (Class 0) and post-2000 hits (Class 1) but had lower performance for pre-2000 hits (Class 2). The lower F1 score for Class 2 suggests room for improvement in distinguishing older hits, potentially due to the reduced availability of distinguishing features in the training data. Despite achieving the highest classification score for pre-2000 hits among models, the neural network had one of the lowest f1-scores for non-hits (Class 0). The neural network's accuracy was also considerably lower than that of XGBoost and Random Forest, which could be due to the model struggling to capture the simpler relationships, or a focus on the minority classes after SMOTE.

Hypertuning:
Preliminary tuning suggested that using a learning rate of 0.01 and training for 30 epochs provided good performance. Different architectures were tested, including a 128-64-32 deeper network, which gave the best results, and simpler 128-64 and 64-32 models, which had slightly worse but not significantly different results, maintaining similar macro F1-scores and classification reports.

Consolidated Results and Interpretation



Metric	Random Forest	XGBoost	KNN (All Features)	Neural Networks
Accuracy	0.858	0.851	0.766	0.811
Precision				
Class 0	0.92	0.92	0.93	0.95
Class 1	0.70	0.70	0.53	0.62
Class 2	0.40	0.37	0.25	0.32
Macro Avg	0.67	0.66	0.57	0.63
Weighted Avg	0.86	0.86	0.84	0.87
Recall				
Class 0	0.90	0.89	0.78	0.82
Class 1	0.77	0.77	0.78	0.84
Class 2	0.42	0.47	0.57	0.69
Macro Avg	0.70	0.71	0.71	0.78
Weighted Avg	0.86	0.85	0.77	0.81
F1-score				
Class 0	0.91	0.91	0.85	0.88
Class 1	0.73	0.73	0.63	0.71
Class 2	0.41	0.41	0.34	0.44
Macro Avg	0.68	0.68	0.61	0.68
Weighted Avg	0.86	0.86	0.79	0.83

Figure 14: Results from all models

Model Performance Analysis:

- Random Forest demonstrated balanced performance across all classes with a macro F1-score of 0.68 and an overall accuracy of 85.8%. It showed strong classification capabilities, particularly for non-hits (Class 0) and post-2000 hits (Class 1), making it robust in handling different song categories effectively.

- XGBoost performed similarly with a macro F1-score of 0.68 and an accuracy of 85.1%. It exhibited slightly better precision for non-hits and post-2000 hits compared to Random Forest but struggled more with pre-2000 hits (Class 2), indicating issues in identifying older hits accurately.
- KNN had the lowest overall performance, with a macro F1-score of 0.61 and an accuracy of 76.6%. It particularly struggled with pre-2000 hits, highlighting significant difficulties in classification, especially for hits. Its lower performance suggests that KNN is less suited for high-dimensional and imbalanced datasets like this one.
- Neural Network showed potential with a macro F1-score of 0.68 and an accuracy of 81.1%. Interestingly, it achieved the best F1-score for pre-2000 hits (Class 2) among all models, but a lower F1-score for non-hits (Class 0) and overall accuracy compared to Random Forest and XGBoost. This initially seems counterintuitive but can be explained by the dataset's imbalance, with non-hits being the dominant class. The network's ability to perform well on the less represented class (pre-2000 hits) indicates that it might have a better capacity to understand the underlying musical characteristics.

The results suggest that while all models benefited from SMOTE in handling class imbalances, Random Forest and XGBoost were the most effective, with the neural network showing potential but requiring further tuning and refinement.

Model Insights:

- **Easier Prediction of Non-hits:** Across all models, it was easier to predict non-hits (Class 0), as indicated by higher precision and recall scores. This is likely due to the larger representation of non-hits in the dataset, allowing models to learn their characteristics better.
- **Relatively Good Performance for Post-2000 Hits:** The models performed relatively well in predicting post-2000 hits (Class 1), with reasonable precision and recall. This suggests that the audio features and metadata available for more recent songs provide clearer distinctions that the models can leverage.
- **Challenges with Pre-2000 Hits:** All models struggled with classifying pre-2000 hits (Class 2). The lower precision and recall scores for this class indicate difficulties in delineating these songs from others. This could be due to fewer samples and more complex temporal trends, making it harder for the models to identify distinguishing features for older hits. The neural network showed the best performance here, suggesting its potential to understand music characteristics better. A potential future direction could involve dropping the release date as a feature to assess how well models can auto-categorize songs into decades purely based on their musical attributes, beyond the current project's scope.

Confusion Matrix Insights:

- The confusion matrices for all models show that none of them categorized any pre-2000 hits as post-2000 hits or vice versa. This could be due to the release date being too influential, which is supported by feature analysis. It indicates that there were not enough trends in the music to properly categorize songs across decades.
- The importance of the release date, as confirmed by the feature analysis, underscores the need for additional audio data, lyrics, and better genre tags to improve classification accuracy. This could allow for more accurate categorization based on the music's intrinsic characteristics rather than temporal context.

Overall, the analysis highlights that while machine learning models can effectively classify non-hits and post-2000 hits, predicting pre-2000 hits remains challenging. This underscores the need for further feature engineering or additional data to improve model performance for older songs.

Interesting and Challenging Aspects

This project presented several fascinating and challenging elements:

- **Surprising Model Performance:** Despite using a dataset with relatively few detailed audio features, the models achieved reasonable classification accuracy. This indicates that features like tempo, key, and energy were sufficiently informative for predicting song popularity.
- **Impact of Genre Tags:** Incorporating genre tags significantly improved model performance. The genre embeddings provided crucial contextual information that enhanced the models' ability to differentiate between hits and non-hits.
- **Predictability of Newer Music:** The models found it easier to predict post-2000 hits compared to pre-2000 hits, suggesting that contemporary music may follow more consistent patterns, indicating a formulaic approach in recent music production.
- **Challenges with High-Dimensional Data:** Handling high-dimensional genre embeddings and ensuring efficient processing required careful tuning and validation, particularly with PCA for dimensionality reduction.
- **Balancing the Dataset:** Using SMOTE to address class imbalances improved recall for minority classes but introduced challenges in generating synthetic samples that accurately represented these classes.
- **Classifying Models in Different Eras**: Classifying models in different eras did not happen in this project, likely because the decade feature was included. However, with more detailed audio features and an initial machine learning pass to effectively auto-classify songs into their respective eras, excluding the decade feature may allow for better cross-classification. This could enhance the models' ability to categorize songs purely based on their musical attributes.

Limitations

- **Dataset Constraints:** The inability to use the Million Song Dataset resulted in missing more detailed audio features. The current dataset has relatively low-dimensional fields, which may not capture the full complexity of the songs. This limitation likely impacted the model's ability to accurately predict song popularity.
- **Data Inaccuracy:** There were instances of inaccurate data within the dataset. For example, some rap songs had low counts of "speechiness," contradicting one of the defining features of the genre. These inaccuracies could have skewed the model's learning and predictions.
- **Lack of Lyrical Information:** Due to rate limits on the Genius API, it was not possible to incorporate lyrical content into the dataset. Including lyrics could have provided additional context and improved the model's ability to predict song popularity.
- **Genre Embeddings Dimensionality:** High-dimensional genre embeddings had to be reduced to avoid an excessively large dataframe, which limited the richness of the genre information that could be used for model training.

- **Computational Limitations:** Running Support Vector Machines (SVMs) on the large dataset was infeasible with only a CPU, as they ran overnight without completing. This prevented the inclusion of SVM results in the analysis, which could have offered additional insights.

Future Work

Given additional time and resources, there are several directions this project could take to further improve and expand the findings:

1. **Integration of the Million Song Dataset:**
    - **Balanced Dataset:** Using the Million Song Dataset with a balanced representation of hits and non-hits would enhance the robustness of the models. This dataset includes more detailed audio features that could capture the complexity of songs more effectively. Furthermore, an initial machine learning pass to effectively auto-classify songs into their respective eras, without including the release year feature may allow for better cross-classification across eras.
    - **Lyrical Information:** Incorporating lyrical data from the Million Song Dataset would provide additional context and improve the models' ability to predict song popularity. Lyrics often play a significant role in a song's appeal and could help in identifying trends and patterns that are not captured by audio features alone.
2. **Utilization of User-Submitted Tags:**
    - **Last.fm Tags:** Including user-submitted tags from platforms like Last.fm could offer a more nuanced categorization of the themes and emotions associated with each song. These tags can capture aspects of music that are not easily quantified by audio features, such as mood, style, and thematic content.
3. **Advanced Model Architectures and Hyperparameter Tuning:**
    - **Neural Networks:** Further exploration of advanced neural network architectures, including recurrent neural networks (RNNs) and convolutional neural networks (CNNs), could improve the capture of temporal and spatial relationships in the data.
    - **Hyperparameter Tuning:** Extensive hyperparameter tuning using techniques such as Bayesian optimization could help in finding the optimal settings for each model, potentially improving their performance.
4. **Incorporation of Additional Data Sources:**
    - **Social Media Trends:** Including data from social media platforms like Twitter and Instagram could provide real-time insights into popularity trends and public sentiment.
    - **User-Generated Content:** Analyzing user-generated content, such as playlists and user reviews, could offer valuable information on how songs are perceived by listeners.
5. **Addressing Computational Limitations:**
    - **High-Performance Computing:** Utilizing high-performance computing resources, such as GPUs or cloud-based services, would enable the training of more complex models and handling of larger datasets, including SVMs and deep learning models.

Future Questions:

Several questions remain unanswered and could guide future research:

- **Temporal Dynamics:** How do the factors influencing song popularity change over time, and can these changes be predicted with greater accuracy using more comprehensive datasets?

- **Cultural Influence:** To what extent do cultural and regional differences impact song popularity, and how can these factors be integrated into predictive models?
- **Interactivity and User Preferences:** How can real-time user interactions and preferences be incorporated into the models to provide more dynamic and personalized music recommendations?

Conclusion

In conclusion, this project successfully applied machine learning models to predict song popularity, identifying the period in which a song would have been a hit. Despite the limitations, the models, particularly Random Forest and XGBoost, showed strong performance in classifying non-hits and post-2000 hits. However, predicting pre-2000 hits remains a challenge, indicating the need for further feature engineering and more comprehensive datasets.

Bibliography

1) Cozar, M. (2022, December 21). Predicting song popularity based on lyrics. Medium. https://medium.com/latinxinai/predicting-song-popularity-based-on-lyrics-fee599165be0
2) GeeksforGeeks. (2023, February 6). XGBoost. https://www.geeksforgeeks.org/xgboost/
3) K, J. (2021, August 30). SMOTE. Medium. https://towardsdatascience.com/smote-fdce2f605729
4) Lavasani, A. (2024, May 8). Classic machine learning in python: K-nearest neighbors (KNN). Medium. https://medium.com/@amirm.lavasani/classic-machine-learning-in-python-k-nearest-neighbors-knn-a06fbfaaf80a
5) Nasreldin, M. (2018, May 14). Song popularity predictor. Medium. https://towardsdatascience.com/song-popularity-predictor-1ef69735e380
6) Pham, J.Q. (2015). Predicting Song Popularity. Stanford University. https://cs229.stanford.edu/proj2015/140_report.pdf
7) What is a neural network?. IBM. (2021a, October 6). https://www.ibm.com/topics/neural-networks
8) What is Random Forest?. IBM. (2021b, October 20). https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,Decision%20trees