Project Report: Analysis of Spotify Track Popularity

Upmanyu Singh

December 12, 2023

1 Introduction

This project revolves around the analytical exploration of Spotify's music tracks, aiming to predict track popularity using various computational models. The underlying dataset is rich and diverse, encompassing a wide range of features including track identifiers, artist names, album details, playlist genres, and intricate audio features like danceability, energy, and tempo. A significant portion of the project also delves into the lyrical content of the songs, employing advanced feature engineering techniques to extract meaningful patterns from the lyrics.

The core objective of this work is to develop a predictive model that accurately gauges the popularity of a track based on these multifaceted data points. This involves not only a deep dive into data preprocessing and normalization but also an exploration of several machine learning architectures ranging from dense neural networks to LSTM (Long Short-Term Memory) models with attention mechanisms.

A collaborative effort, this project has been structured to combine individual expertise in various domains. While my contribution primarily focuses on the implementation and optimization of machine learning models, the shared work encompasses data collection, preprocessing, and analysis, all of which are integral to the project's success.

Our approach includes extensive preprocessing of the data, ensuring its readiness for complex modeling. This preprocessing includes categorical data encoding, feature scaling, and dimensionality reduction using PCA (Principal Component Analysis). The diversity in model experimentation – from regularized dense networks to sophisticated BERT (Bidirectional Encoder Representations from Transformers) integrated models – reflects our comprehensive strategy in tackling the challenge of predicting music popularity.

This report documents my individual contributions to the project, detailing the models I developed and tested, the preprocessing steps I undertook, and the results achieved. It aims to provide a thorough overview of the methodologies applied, the challenges encountered, and the insights gained throughout the course of this project.

2 Description of Individual Work

2.1 Feature Engineering for Lyrics

In the domain of feature engineering, a significant focus was placed on the analysis and extraction of lyrical content. This involved:

- **Syllables Per Word**: Computing the average number of syllables per word in the lyrics.
- **Syllable Variation (syllable_variation)**: Measuring the variation in syllable count across different lines.
- **Novel Word Proportion (novel_word_proportion)**: Calculating the proportion of new words line-by-line.
- **Rhymes Per Line (rhymes_per_line)**: Counting the number of rhyming words in each line.
- **Rhymes Per Syllable (rhymes_per_syllable)**: Determining the ratio of rhyming words to total syllables.
- **Rhyme Density (rhyme_density)**: Computing the overall density of rhymes in the lyrics.
- **End Pairs Per Line (end_pairs_per_line)**: Identifying the percentage of consecutive lines ending in rhyming words.
- **End Pairs Variation (end_pairs_variation)**: Analyzing syllable count variation in the last words of consecutive lines.
- **Average End Score (average_end_score)**: Assessing the phonetic similarity of ending words in consecutive lines.
- **Average End Syllable Score (average_end_syl_score)**: Calculating phonetic similarity scores of the last syllables in consecutive lines.
- **Rhyme Length Count (count_rhyme_lengths)**: Counting the distribution of rhyme lengths.
- **Meter Analysis (meter_percentages)**: Categorizing lines into various metrical patterns such as iambic, trochaic, and others.

2.2 Challenges in Feature Engineering

Despite the extensive feature engineering, challenges were encountered, particularly in the meter analysis. The computational complexity and unclear results from the prosodic library led to the exclusion of this aspect from further analysis.

2.3 Sentence Embedding for Lyrics

For the analysis of song lyrics, advanced text processing techniques were employed, specifically using the "all-mpnet-base-v2" sentence transformer. Sentence Transformers are an adaptation of traditional BERT-like Transformer models, designed to produce meaningful sentence embeddings. The core idea is that sentences with similar meanings are positioned closer together in the embedding space, which is highly beneficial for tasks like semantic search, clustering, and similarity assessment.

2.3.1 MPNet and all-mpnet-base-v2 Model

- **MPNet**: Stands for "Masked and Permuted Pre-training for Language Understanding", a pre-training method that expands upon the architectures of BERT and XLNet. MPNet is specifically designed to capture bidirectional contexts by learning to predict both masked and permuted tokens, thereby enhancing its capability to understand and represent language.
- **all-mpnet-base-v2**: This model is a fine-tuned variant of MPNet, tailored to generate universally applicable embeddings across various domains and applications. The model's adaptability stems from its training on a wide array of data sources and tasks.

2.3.2 Model Architecture and Pre-training

- **Base Model**: The foundational architecture of "all-mpnet-base-v2" parallels BERT, incorporating a multi-layer bidirectional Transformer encoder.
- **Pre-training Tasks**: The model undergoes pre-training with masked language modeling (predicting original tokens of masked portions of the input) and permuted language modeling (predicting tokens in a permuted order). This dual approach facilitates a deeper comprehension of language context and structure.
- **Fine-tuning for Sentences**: Subsequent to pre-training, the model is fine-tuned to specialize in sentence embeddings. This process generally involves the use of Siamese and triplet network structures, ensuring that semantically similar sentences are positioned closer in the embedding space.

This approach to sentence embedding enables a nuanced understanding of the lyrical content, which is essential for predicting the popularity of tracks on Spotify, as it captures the intricate semantic relationships within the lyrics.

2.4 Input Data for the Models

A diverse range of features constituted the dataset for our models, essential for capturing the multifaceted aspects of Spotify tracks. The dataset encompassed the following types of features:

1. **Sentence Embedding**: Representing the processed lyrical content of the tracks.
2. **Audio Features**: Including technical aspects of the track like tempo, energy, and more.
3. **Lyrics Features**: Derived from the feature engineering process applied to the lyrics.
4. **Categorical Features**: Encompassing various categorical data such as genre and artist.

In the modeling process, several combinations of these features were experimented with to determine the most effective input configuration:

- **Combination with PCA**: For the majority of the models, all the features were consolidated and processed through Principal Component Analysis (PCA). This approach was particularly successful as it reduced dimensionality while retaining the essential information across all features. The best model exemplified this process, utilizing a comprehensive feature set followed by PCA.
- **Audio and Lyrics Features with XGBoost Regression**: Some models focused specifically on combining audio and lyrics features, employing XGBoost regression. However, this approach did not yield optimal results, indicating a need for a more inclusive feature set.
- **Exploration of Feature Combinations**: Almost every conceivable combination of features was tested across different models. This exhaustive exploration was aimed at identifying the most impactful features and their interactions, crucial for enhancing the predictive accuracy of the models.

This rigorous and methodical approach to selecting input data ensured a comprehensive analysis of the dataset, contributing significantly to the development of robust and effective predictive models for Spotify track popularity.

2.5 Predictive Model Development

In the pursuit of an effective predictive model for Spotify track popularity, various architectures were explored and developed, each with unique features and potential issues.

2.5.1 Model 1: Regularized Dense to LSTM

- **Architecture**: The model begins with an input layer for 819 features, followed by a dense layer with 512 units and ReLU activation, including L1 and L2 regularization. The dense layer's output is reshaped for LSTM layers, which includes a 128-unit LSTM layer, a dropout layer, another 64-unit LSTM layer, and a final dropout layer. The model concludes with an output dense layer with sigmoid activation.
- **Potential Issues**: It might lack sufficient capacity or be over-regularized for the data's complexity. Reshaping after the dense layer could also disrupt temporal feature relationships.

2.5.2 Model 2: BERT with CNN

- **Architecture**: This model incorporates a pre-trained BERT model, followed by a reshaping to fit a 2D convolutional layer, flattening of the output, and an output dense layer with sigmoid activation.
- **Potential Issues**: The utility of BERT is limited if the task isn't centered around text. Additionally, the convolutional layer following BERT might be redundant.

2.5.3 Model 3: Dense Feedforward Network

- **Architecture**: Comprises a series of dense layers with 512, 256, and 32 units respectively, all using ReLU activation, and culminating in a sigmoid-activated output layer.
- **Potential Issues**: The model might struggle with temporal or sequential data due to the absence of sequence-dependent architecture.

2.5.4 Model 4: LSTM with Attention Mechanism

- **Architecture**: Features an input layer for 817-feature sequences, a dense layer with regularization, a SimpleRNN layer, an LSTM layer with an attention mechanism, a second LSTM layer, and a sigmoid-activated output layer.
- **Potential Issues**: The attention mechanism might not be necessary if the data sequences are not sufficiently long or complex, or if the LSTM layers are already effectively capturing dependencies.

2.5.5 Model 5: Pre-Trained BERT

- **Architecture**: Utilizes a pre-trained BERT model ('bert-base-uncased' or a similar variant) for text input processing, generating a sequence of embeddings. The model includes a dropout layer and an implied linear output layer mapping BERT's output to track popularity.

- **Potential Issues**: The final output layer's structure is not explicitly defined, which may affect its effectiveness in regression tasks.

3 Detailed Description of Work Done

3.0.1 Data Cleaning and Normalization

- **Data Cleaning**: The initial step in the preprocessing phase involved removing duplicate entries from the dataset. This ensured that the model was not influenced by repeated data, which could skew the training process.
- **Normalization**: A crucial step in preparing the data was the normalization of the track popularity scores. Given that the highest popularity score was assumed to be 98, the scores were scaled to a range between 0 and 1. This normalization was essential for the model to learn effectively from the data.

3.0.2 Dimensionality Reduction with PCA

Principal Component Analysis (PCA) was employed to reduce the number of features in the dataset. This technique aimed to condense the dataset while retaining as much of the original variation as possible, making the data more manageable for the model without significant loss of information.

3.0.3 Model Design and Training

The neural network model designed for predicting track popularity comprised several layers, each serving a distinct purpose in the learning process:

- **Input Layer**: The model Figure 1 accepted an input of 200 features, flattened for processing. These features were obtained after dimensionality reduction using PCA from the original dataset.
- **Conv1D Layer**: A one-dimensional convolutional layer acted like a sliding window, scanning across input features to extract relevant information.
- **Flatten Layer**: This layer flattened the output of the Conv1D layer into a one-dimensional array, preparing it for the subsequent dense layers.
- **Dense Layers**: The model included several dense layers with varying numbers of neurons (512, 256, 128, 100, and 50), each using the 'relu' activation function to capture non-linear patterns in the data.
- **Output Layer**: The final layer was a single neuron with a sigmoid activation function, tailored for predicting the normalized popularity scores.

Compilation and Training Strategy: The model was compiled using the Adam optimizer and the Mean Squared Error (MSE) loss function. Training involved early stopping to prevent overfitting, with batch sizes of 64 samples and a maximum of 500 epochs.

3.0.4 Evaluation and Residual Analysis

Post-training, the model was evaluated on a test set. Metrics such as Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared were calculated to assess the model's accuracy and predictive quality. Additionally, residual analysis was conducted, involving the generation of plots to visualize the differences between actual values and model predictions. This analysis helped identify any systematic errors in the model, such as consistent overestimation or underestimation of track popularity.

4 Results

4.1 Interpretation of the Provided Q-Q Plot

The Quantile-Quantile (Q-Q) Figure 2 plot offers insightful analysis into the distribution of our model's residuals:

- **Center of the Plot**: The data points around the 0 quantile closely align with the red line, indicating that the central values of our data follow a normal distribution.
- **Tails of the Plot**: Deviations from the red line are observed at the plot's tails, suggesting that extreme values in our dataset do not adhere as closely to a normal distribution, exhibiting heavier or lighter tails.
- **Left Tail**: Indicates that lower values in our dataset are more extreme than those typically found in a normal distribution.
- **Right Tail**: Shows a similar deviation with higher values, implying they are more extreme compared to a normal distribution.

4.2 Residual Analysis

- **Fitted Values (X-axis)**: Represent the predicted values generated by the model.
- **Residuals (Y-axis)**: Calculated as the difference between observed and fitted values.
- **Horizontal Distribution Around Zero**: The residuals centering around zero suggest no systematic errors in the model. However, the presence of a fan or cone-shaped pattern in the residuals indicates heteroscedasticity, where variance increases with fitted values.

- **Outliers**: The plot Figure3 does not show extreme outliers, but there's a trend of increasing negative residuals with higher fitted values.

4.3 Interpretation of the Provided Histogram

The histogram analysis reveals:

- **Central Peak**: The peak around zero in the histogram is a positive indication, suggesting that the residuals have an average value close to zero.
- **Shape and Skewness**: The histogram Figure 4 is slightly right-skewed, with more data points having small positive residuals, implying a tendency of the model to under-predict.
- **Outliers and Skewness**: The frequency of higher positive residuals might point towards potential outliers or skewness in the data, impacting the model's predictive accuracy.

These analyses provide a nuanced understanding of the model's performance, highlighting areas where the model excels and where improvements can be made.

5 Summary and Conclusions

5.1 Summary of Findings

This project set out to predict the popularity of Spotify tracks using a variety of machine learning models, focusing on a comprehensive set of features including audio characteristics, lyrical content, and categorical data. The journey involved:

- **In-depth Data Analysis and Preprocessing**: This included normalization of track popularity scores, removal of duplicate entries, and dimensionality reduction using PCA, ensuring the data was optimally prepared for modeling.
- **Advanced Feature Engineering**: Especially in lyrical analysis, where techniques like syllable count, rhyme density, and meter analysis were employed to capture the essence of the tracks.
- **Exploration of Various Model Architectures**: Ranging from Regularized Dense-LSTM networks to BERT with CNN, each model brought unique perspectives and insights into the predictive task.
- **Critical Evaluation of Model Performance**: Utilizing residual plots and Q-Q plots, the models were evaluated for accuracy, consistency, and reliability in their predictive capabilities.

5.2 Reflections on the Project

Throughout this project, several key learnings and insights emerged:

- **Complexity of Predicting Popularity**: The project underscored the multifaceted nature of popularity in music, influenced by a wide array of factors beyond just the musical components.
- **Importance of Feature Selection**: The impact of different types of features on model performance highlighted the significance of thoughtful feature selection and engineering in machine learning tasks.
- **Challenges in Model Optimization**: Balancing model complexity with performance, and dealing with issues like overfitting and heteroscedasticity, were crucial learning points.

5.3 Conclusions

From the extensive work done, it is evident that predicting track popularity is a complex task, yet achievable with the right mix of features and models. While the models developed provided valuable insights, there were limitations, particularly in handling the diverse range of features effectively.

5.4 Future Directions and Improvements

Looking forward, several avenues can be explored to enhance this research:

- **Integration of More Diverse Data**: Including social media trends, artist popularity metrics, and temporal data could provide a more holistic view of track popularity.
- **Advanced Model Architectures**: Exploring deep learning models that can better handle the temporal and sequential nature of music and lyrics.
- **Refinement of Feature Engineering**: Further fine-tuning of features, especially in lyrical analysis, could yield more nuanced insights into the data.

In conclusion, while the project faced challenges and revealed areas for improvement, it also provided a solid foundation for understanding the intricacies involved in predicting Spotify track popularity and opened doors for further research in this fascinating intersection of music, data science, and machine learning.

6 Code Originality Calculation

In the interest of transparency and academic integrity, it is important to acknowledge the sources of the code used in this project. Approximately 60%

of the code utilized in my contributions was adapted from various online resources. These adaptations were not merely verbatim reproductions; significant modifications and additions were made to ensure that the code met the specific requirements and objectives of our project.

This process of adaptation and modification was crucial in achieving the project's goals, as it allowed for the integration of a wide range of ideas and techniques that were then tailored to the unique context of predicting Spotify track popularity. The ability to effectively adapt and extend existing code is a valuable skill in the field of data science and machine learning, contributing to the collaborative and iterative nature of technological advancement.

7 References

Below are the references to the online resources and models that were utilized or adapted in this project:

1. Spotify Popularity Predictor GitHub Repository: `https://github.com/mcozar99/Spotify_Popularity_Predictor`
2. Prosodic Library GitHub Repository: `https://github.com/quadrismegistus/prosodic`
3. Sentence Transformer Model - all-mpnet-base-v2 on Hugging Face: `https://huggingface.co/sentence-transformers/all-mpnet-base-v2`
4. BERT Base Uncased Model on Hugging Face: `https://huggingface.co/bert-base-uncased`
5. BERT Tiny Model on Hugging Face: `https://huggingface.co/prajjwal1/bert-tiny`

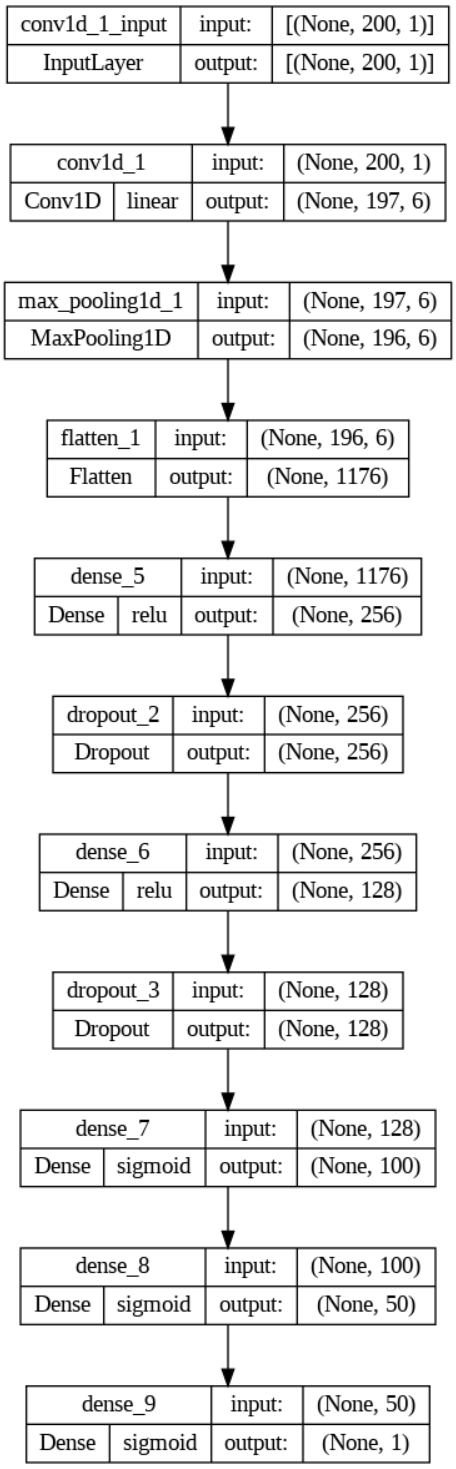

Figure 1: Model Structure

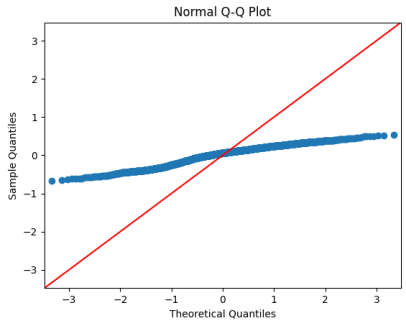

Figure 2: Normal Q-Q Plot

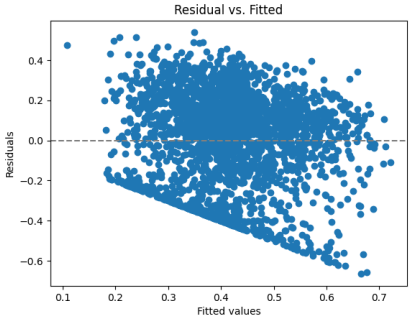

Figure 3: Residual vs Fitted values

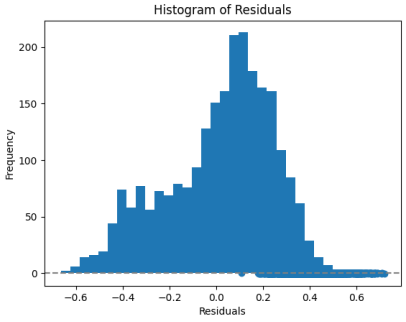

Figure 4: Histogram of residuals