# Lyrics Genie 🧞

## Predicting Song Popularity
## &
## Building Artist-Specific Lyric Generators


Team 5

Ji Woo Suh

Sanjana Godolkar

Upmanyu Singh


# Introduction

Lyrics Genie is an innovative tool that predicts song popularity using NLP.

Our approach involves classification and regression models with advanced NLP techniques to analyze and generate lyrics. We then predict the potential popularity of these newly created lyrics. Our project is at the intersection of technology and music, aiming to forecast hit songs and offer insights into the elements that contribute to their success.

# Regression

# Data Description

| | | | |
|---|---|---|---|
| track_name | track_album_release_date | energy | instrumentalness |
| track_artist | playlist_name | key | liveness |
| | playlist_genre | loudness | valence |
| **track_popularity (target)** | playlist_subgenre | mode | tempo |
| track_album_name | lyrics | speechiness | duration_ms |

- **Data Range and Processing: The dataset's target variable ranges from 0 to 100. For analysis, a two-different approach is employed: first, regression is performed, and then classification.**
- **Data Cleaning and Selection: The initial dataset contained 18,000 rows. After filtering out non-English songs to retain only those with English lyrics, 12,170 rows remained for analysis.**

# New Feature Creation

- syllable_per_line: The average number of syllables per line of lyrics.
- syllable_per_word: The average number of syllables per word within the lyrics.
- syllable_var: The variation of syllable counts across the lyrics.
- novel_word_proportion: The proportion of unique words in the lyrics.
- rhymes_per_line: The average number of rhymes per line.
- rhymes_per_syllable: The average number of rhymes per syllable.
- rhyme_density: The density of rhymes within the lyrics.
- end_pairs_per_line: The frequency of paired rhymes at the end of lines.
- end_pairs_variation: The variation of paired rhymes at the end of lines.
- average_end_score: The average score of ending rhymes in the lyrics.
- average_end_syl_score: The average syllable score of ending rhymes.
- count_rhyme_lengths: The count of different rhyme lengths in the lyrics.



Reference paper: https://cs.uwaterloo.ca/~browndg/CMMR15data/CMMR2015paper.pdf
**(Can Song Lyrics Predict Hits?)**
Python library: pronouncing (for rhymes) and syllapy (for syllables)

# Feature Engineering

1. **Categorical Encoding**: Converted 'playlist_genre', 'playlist_subgenre', and 'track_album_release_date' into numerical formats using one-hot and label encoding for machine learning readiness.
2. **Feature Expansion**: Augmented the dataset with one-hot encoded artist and genre features to better capture categorical information.
3. **Sentence Embedding**: Applied a pre-trained model (all-mpnet-base-v2) to translate lyrics into numerical embeddings, enhancing the dataset with semantic text features.
4. **PCA:** We did the dimensionality reduction and reduced the features to 200 (optimum point).

# Tried Models

1. RNN + LSTM + Attention
2. RNN + LSTM
3. MLP
4. BERT + CNN
5. BERT + MLP
6. LSTM
    a. Tried removing all the combination of audio features (valence, tempo, acousticness, etc), lyrics features (syllables per line, syllables per word etc) and sentence embedding.
    b. Tried different embedding techniques:
        i. One of the interesting technique we tried is creating Covariance matrix of glove word to vector and then apply CNN to it.

Almost all the models tried were giving negative R-square.

# Model Architecture

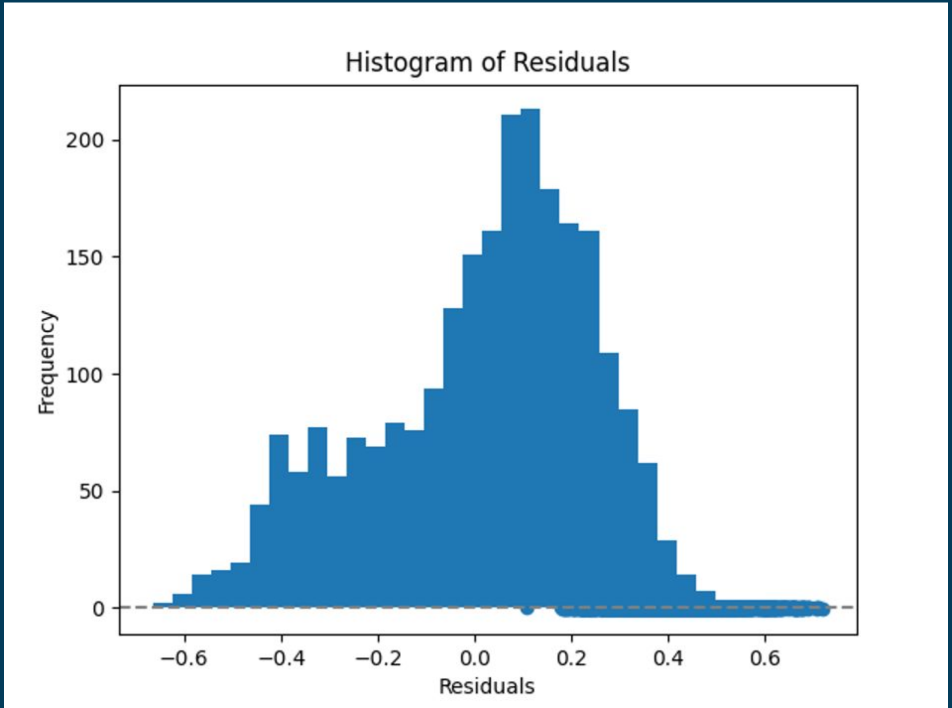
Model: "sequential_1"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv1d (Conv1D) | (None, 197, 6) | 30 |
| max_pooling1d (MaxPooling1D) | (None, 196, 6) | 0 |
| flatten (Flatten) | (None, 1176) | 0 |
| dense (Dense) | (None, 256) | 301312 |
| dropout (Dropout) | (None, 256) | 0 |
| dense_1 (Dense) | (None, 128) | 32896 |
| dropout_1 (Dropout) | (None, 128) | 0 |
| dense_2 (Dense) | (None, 100) | 12900 |
| dense_3 (Dense) | (None, 50) | 5050 |
| dense_4 (Dense) | (None, 1) | 51 |

Total params: 352239 (1.34 MB)
Trainable params: 352239 (1.34 MB)
Non-trainable params: 0 (0.00 Byte)

Our model uses a convolutional neural network designed for regression to predict music track popularity, incorporating layers with ReLU and Sigmoid functions for complexity and using Mean Squared Error and Adam Optimizer for learning. Key settings for training are a 0.0001 learning rate and batch size of 64.

# Evaluation of Model




- The residuals are assumed to be following the normal distribution.
- The other evaluation metrics on hidden test dataset are:
- MSE loss: 0.049
- MAE loss: 0.1702
- R-squared: 0.23
- Adjusted R-Squared: 0.17

# Classification

# Data Pre-processing

Target Variable Processing:

Transformed 'track_popularity' into four classes (0-3) based on popularity scores.

0: Low Popularity (0-26)

1: Avg Popularity( 27-47)

2: Mid Popularity (48-61)

3: High Popularity (62-100)

```
count    13030.000000
mean        42.591097
std         24.292148
min          0.000000
25%         26.000000
50%         47.000000
75%         61.000000
max        100.000000
Name: track_popularity, dtype: float64
```

# Embedding

BERT Embeddings:

- Utilized 'bert-tiny' model for generating embeddings from lyrics.
- Embeddings provide deep contextual representations of text data.

Feature Combination and PCA:

- Combined BERT embeddings with other preprocessed features.
- Applied PCA (Principal Component Analysis) to reduce dimensionality, retaining 200 components.

# Model Architecture

Model Choice:

- XGBoost, an advanced and efficient implementation of gradient boosting.

Model Configuration:

- Objective: 'multi:softmax' for multiclass classification.
- Hyperparameters: Learning Rate (0.1), Max Depth (3), Estimators (200), Subsample (0.8).

Training Process:

- Model trained on PCA-reduced feature set from the preprocessed dataset.
- Emphasis on handling multi-class classification through softmax objective.

Advantages of XGBoost:

- Handles a large number of features efficiently.
- Robust to overfitting and capable of capturing complex patterns.

# Evaluation and Result Analysis

Test Set Accuracy: 0.387434554973822
XGBoost Model saved successfully.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.42 | 0.36 | 0.39 | 315 |
| 1 | 0.37 | 0.45 | 0.41 | 365 |
| 2 | 0.33 | 0.24 | 0.28 | 341 |
| 3 | 0.42 | 0.51 | 0.46 | 316 |
| accuracy | | | 0.39 | 1337 |
| macro avg | 0.39 | 0.39 | 0.38 | 1337 |
| weighted avg | 0.38 | 0.39 | 0.38 | 1337 |

Model Evaluation:

- Accuracy metric used to evaluate performance: Achieved an accuracy of 38.7% on the test set.

Interpreting Results:

- Track Classification: The model shows promise in classifying songs into popularity classes, providing a foundation for further development.
- Targeted Improvements: Identifying areas where the model performs well and where it falls short allows for targeted improvements to enhance classification accuracy.

Model Saving and Deployment:

- XGBoost model saved as 'xgb_model.json' for future use.
- PCA model saved as 'pca_model.joblib' to maintain feature transformation consistency.

# App Demo