Sentiment Analysis and Explicit Classification of Songs from Lyrics

Adrian Perez, Maxim Yurkovsky, Divleen Lota
Aidan DeVaney, Danny Yu, Taylor Ueda
Cory Pham, Arezoo Ghasemzadeh

Abstract—**Music is a popular form of entertainment that allows individuals to creatively express their own realities and experiences into a work of art. However, this expressionism could lead to problems when it comes to protecting children from potentially harmful content. Parents are unable to monitor what their children view at all times and most content can only be judged after the fact, meaning children would already be exposed to harmful topics. Natural Language Processing provides a solution to this issue, converting words into processable numeric data. Our model will help filter out content to make social media safer for kids, which would make parents less distressed from over-monitoring their children. The current methodologies that the primary music streaming platforms have created(Spotify for Kids, etc) are curated lists of music, instead of comprehensive algorithms that actual attempt to screen the content and context of songs. While parents can disallow their children using music labeled 'explicit', this is only half the battle. There are is a large percentage of songs that do not contain explicit lyrics, but their subtext is highly inappropriate for children. Our best solution is our LSTM model, which performs high level sentiment analysis, and determines whether songs are truly explicit or not. By utilizing the LSTM model, it was able to read the lyric sequence from beginning to end as well as end to beginning through biderection and memory gates. As a result, while the accuracy was lower, we ultimately chose the LSTM out of the other models that we trained and tested because it had the least change of over fitting. Furthermore, the LSTM model had a greater ability to understand slang and uncover the context of explicit song ideas, even when the words themselves might be labeled as clean, which an essential aspect for classifying the songs into their correct category.**

I. Introduction

Within human society, our main forms of communication are through language. Through arbitrary symbols and structure, people are able to convey their messages and have people understand how they feel. These symbols and structures come from a variety of places, and the origins of said language is highly rooted within our cultures. Due to the diversity of these forms of communication, it can be difficult to track down what certain words mean and different interpretations could happen on the same phrase. New methods of machine learning were utilized to mitigate this issue. This includes using artificial neural networks to discover semantics in words, using these meanings to form correlations between different phrases [1]. Natural Language Processing can also be used to further analyze text data to be able to place sentiment on words. Further improvements can be made when these techniques are combined with Naive Bayes and Support Vector Machines (SVMs) to categorize content into certain emotions. This is helpful because it allows people to find different genres for various content depending on their mood. [2]

Even though these analyses are useful when trying to understand human language and emotion, we would like to expand on this idea and use sentiment analysis to determine what sort of content is safe for children or not. This classification of "explicit" or "not explicit" allows sentiment analysis to be utilized as a safeguard for children, preventing them from being exposed to potentially harmful topics. With the rise of various music streaming platforms and individualized playlists, there is a pressing need to safeguard younger audiences from inappropriate content. Manual checking of millions of tracks would be far too time-consuming and impractical, so using machine learning to classify explicit content would help automate and increase efficiency and safety for music applications when it comes to content moderation. This would allow users to create safe and family-friendly playlists, make quick analyses on the content of new song releases, and find connections between certain lyrics across different genres, musicians, and time periods.

In this project, we implemented a data pipeline to process words from song lyrics and used Natural Language Processing (NLP) to determine what tracks are age-appropriate for children. First, we preprocessed the lyric data by deleting special characters to remove noise and tokenized the lyrics from the dataset into processable parts. Then, we fed the cleaned lyrics with words into numbers using a vectorizer so that our model can analyze the data. Our cleaned lyrics dataset was then trained on multiple models to see what techniques were effective in classifying song lyrics as explicit or non-explicit. These results helped demonstrate which models would be best suited for our goal of age-appropriate categorization.

Our dataset consisted of various tracks from Spotify playlists full of children and adult songs. We then trained our model using multiple machine learning strategies such as logistic regression, support vector machines (SVMs), random forests, and long short-term models (LSTMs). Our models ended up classifying song lyrics with a range from 77% to 84% accuracy. Since the LSTM model is an RNN that processes sequential data with components such as long-term memory and a gating mechanism that allowed for a bidirectional approach, this made the LSTM model more favorable due to how more efficient it is when reading lyrics and how it can understand slang and uncover the context of explicit song ideas. Although determining what constitutes content as age-appropriate can be subjective due to a lack of a criteria, our classification system, especially through the LSTM model, shows that given any track, there will be an adequate estimation of whether the song contains any explicit content or not. This can be utilized to help monitor children from being exposed to harmful themes and can give relatively accurate predictions on recently generated or unknown tracks assuming their lyrics are provided. Future work could include deployment of our model for live content moderation, more personalized subjective classification based on genre and context, and integration of deeper learning models to reduce bias and form more informative understanding of song lyrics.

II. Methods

A. Finding a Dataset

We initially proposed a model that was going to classifiy whether a track was appropriate for children based off the following input features from Spotify's Audio Features API:

- "Danceability" was going to be a number from 0.0 to 1.0 that measured how suitable a track was for dancing.
- "Loudness" was going to be a floating point number that measured how loud a track is.
- "Speechiness" was a number between 0.0 to 1.0 that measured how prominent spoken words were in a sound track.

We also wanted to use the lyrics of a track, but these had to be queried using the Lyric Genius API since Spotify's API does not provide lyrics. Our target variable was going to be "Valence", another one Spotify's Audio features. Valence was a number between 0.0 and 1.0 measuring the "positivity" of a soundtrack. Our initial idea was that the more positive a soundtrack sounded, the more likely it was that the track was suitable for children. Still, it is a very possible that a track can be positive without being intended for children as well as also be negative while still being made for a young audience. To make matters worse, the Spotify API had unfortunately been deprecated early in 2025. Thus, it would be impossible to query audio features for new tracks.

There were two routes we could take. We could either train a model using Kaggle datasets that contain information collected prior to the API's deprecation or we could build our own dataset by compiling tracks that were identified as being intended for children and tracks that are inappropriate for a young audience. Either route had its tradeoffs. If we choose the first option, we have access to a much larger range of data. However, the model would be unable to predict Valence because the audio features needed are no longer accessible. The concern about Valence mentioned earlier was also a factor in deciding which route we should take. If we were to choose the second option, we would have a smaller dataset but we would have much more control over the classification of the tracks. This model would not rely on any of Spotify's Audio Features. Hence, we decided to follow through with the second option of creating our own dataset.

B. Building A Dataset

Our goal was to have a dataset with a couple thousand tracks and four columns per track. These columns were, "Artist", "TrackName", "Lyrics", and "AgeAppropriate". The first three columns are regular text and, last column is a boolean of 0 or 1. If a track is labeled with a 1, then it is considered to be appropriate for young audience. If the column contains a 0, then it is considered to not be appropriate for child audiences.

To build our dataset, we searched through Spotify for playlists tailored towards a young audience. We understand that not all playlists will be perfect but for now we just wanted to get as many children's tracks as possible. We also followed a similar procedure when looking for tracks that were for adults. However, there was less of a concern that a children's song would be found in an adult playlist. We tried to get as wide a range as possible when it came to adult tracks. That is to say, we looked for some playlists where the tracks were very obviously not meant for children and other playlists that contained tracks where it was not so obvious to tell that the track is not for children. This way we our model could take into consideration tracks that reference topics like sex or drugs, without explicitly talking about them.

Once we knew what playlists we were using, we built a python script to query Spotify for the tracks in each playlist. With the track details, like track name and artist, we queried the Lyric Genius API to get the lyrics. If the track came from one the children's playlists, then it was written into a CSV containing only kids' tracks with a 1 in the AgeAppropriate column. In the other case, the track was written to an adult-only CSV with a 0 in the AgeAppropriate column.

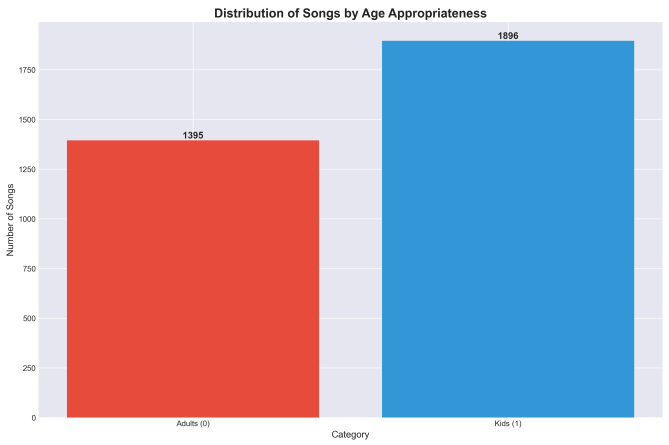


Fig. 1. Class distribution in our dataset

C. Data Pre-Processing

To ensure that the child appropriate CSV was 100% safe and contained no profanity, we utilized a Python library called "better_profanity" that double checked the CSV and eliminated tracks that contained any sexual or inappropriate language. We wanted to ensure that we could get very reliable testing data that was an accurate representation of child-appropriate music and adults-only music.

Next, we had to ensure that the lyrics for each song only contained English words and letters given that our model would only work on primarily English songs. We used the "cleantext" and "unidecode" libraries to ensure that all non-English language structures were properly translated to an existing ASCII character or completely removed. For example, "á" would be translated to "a" while words in non-Latin letters would be removed from the lyrics. This is best for the model, as it would analyze the lyrics better if only English words were present.

Once the children's tracks were fully sanitized as well and all tracks were removed of non-English characters, we merged all of the queried tracks together into one massive CSV and randomly shuffled the data set to create good randomization for the model. Together, there are a total of 3,300 tracks that were used to build the lyrics dataset. This finalized data was then sent over to the modeling group to be processed.

Before the data could be fed into one of our models, we first had to tokenize the data using the Spacy library in Python. After the data was tokenized, we processed the tokens into Term Frequency-Inverse Document Frequency using Sklearn's feature extraction API. We chose TD-IF vectors over Word2Vec, another popular NLP technique, because TD-IF vectors have been shown to be more accurate for classification tasks involving smaller and more controlled datasets [3]. This makes TD-IF a better fit for our data.

D. Models and Algorithms

a) *Support Vector Machines (SVMs):*

Support Vector Machines are popular in the world of machine learning because they are able to perform well for data sets containing many dimensions. Hence, they have been widely popular in areas of Natural Language Processing and Text Classification.

The algorithm's main focus revolves around the hypothesis space and loss function[4]. While there are various choices for a hyperplane, we chose to use a hyperplane in the form of

$$f(x) = wx + b. \tag{1}$$

Where w and b are the parameters that make up the hyperplane. We use a linear hyperplane under the assumption that our data is linearly separable. The goal of the SVM algorithm is to find the optimal hyperplane. In our case, the optimal hyperplane is the one that best classifies the lyrics into the categories of age-appropriate and not age-appropriate. Since our goal is one of classification, our loss function should be in the form of

$$\text{Loss} = -yf(x) \tag{2}$$

where y is the label and $f(x)$ is the predictor function[4]. However, for our SVM is Squared Hinge Loss is used because it is differentiable and enables more efficient parameter changes during training [5].

$$\text{Loss} = \max(0, 1 - yf(x))^2 \tag{3}$$

The squared term enables the algorithm to make larger adjustments to parameters when the loss is large and minor tweaks when the loss is smaller.

Many of the lyrics in our dataset are long and complicated. Combined with NLP techniques to vectorize the lyrics, we can expect that our vectors are going to have a high dimensionality. We hope that this high dimensionality enables the model to make small distinctions between different songs, which will result in a correct classification. Hence, we expect that an SVM model will perform very well after being trained on the dataset.

b) *Logistic Regression:*

Logistic Regression is a useful method of classification in machine learning due to to their probabilistic properties. The algorithm essentially outputs a value between 0 and 1 depending on the various feature values of the dataset. This method is preferred over certain methods such as linear regression due to its resistance to outliers. Its simplicity and efficiency makes it helpful for categorizing data into certain groups, especially if it is for binary classification[6].

The main notation for logistic regression is the sigmoid function, $\sigma(x)$, where its equation is given as

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

where x represents the input score, usually determined after learning the weights and biases from feature values. The algorithm is continually learning from these weights through a process called gradient descent, in which new weights are found by finding the derivative of the loss function with respect to the weights. The loss function can be given as the *Residual Sum of Squares (RSS)* given by

$$\text{RSS} = \sum_{i=1}^{m} \left(y^{(\text{i})} - w^T x^{(\text{i})} \right) \tag{5}$$

where m represents the number of samples, y represents the actual value, w represents the weight vector, and x represents the input values for that feature. This process helps minimize the error between the predicted and actual output value. In our model's case, our logistic regression optimization comes from a approximation of batch gradient descent called L-BFGS[7].

Although logistic regression works better on numeric values, we can circumvent the problem by turning our text data into numerical data through tokenization and making vector values for each unigram and bigram to be used in our model. We allow up to 20 values for our vector and use common english vocabulary as our stop words to prioritize more relevant and significant words. Due to our goal of labeling tracks as "explicit" or "non-explicit", logistic regression should work well here due to its probabilistic values of 0 and 1, and how they can used to perform binary classification.

c) *Random Forest:*

Random forests are an ensemble of smaller models called "decision trees" in which various tress are created and branched off through feature values and classification happens on the leaves or terminal nodes of each tree. The results are then compiled and decided generally upon majority voting in which the category that was chosen by the most trees generally has the highest probability of being the final prediction.

For splitting the features, we choose the features that end up minimizing the impurity of further nodes down on the tree. In our model, we utilize the Gini Impurity equation given by

$$\text{Gini} = 1 - \sum_{i=1}^{n} (p_i)^2 \tag{6}$$

where C represents the number of classes and p_i is the probability of data points that is part of class i.

Choosing the best features grants us the greatest Gini gain, meaning we are able to separate the data points that are of different classes and move closer to homogenized sets, where the purity is higher and we have a better indication of what feature combinations represent what class. This equation can be generalized to

$$\text{Gain} = \text{Gini(parent)} - (w_L \ (\text{Gini(L)}) + w_R \ (\text{Gini(R)})) \tag{7}$$

where w represents the ratio between samples of the children node and the parent node. These weights are multiplied with the Gini impurities of their respective node, added together with other children nodes and subtracted from the Gini impurity of the parent node to calculate the information gained from splitting based on a certain feature.

In our model's case, we had our random forest create 100 random decision trees on our cleaned lyric dataset. We then outputted the final classification result, "age-appropriate" or "not age-appropriate", based on a "soft voting" system, where the class that is chosen is the one with the highest average probability in the ensemble. This allows for the model to become more robust because it is able to capture uncertainty in its predictions rather than being just a black and white decision.

We expect random forests to do well in this classification due to their strong feature selection, non-linear approximation and generalization properties, which make it less prone to overfitting[8]. This is helpful for our classification because our text data is converted into arbitrary vector values which might not always be linear. It is also more efficient when it comes to extracting information from a large number of datasets to formulate more accurate predictions[8]. This is important when we deal with music-streaming text data, as there are a large number of tracks being produced rapidly, and their feature extraction can be more practical when it comes to providing a rough estimate of how much a track is "age-appropriate".

d) *LSTM:* An LSTM (Long Short-Term Memory) model is essentially a type of recurrent neural network that learns patterns in sequences of data. This includes text, speech, time-series, etc., where the order of information matters.

In modern machine learning, LSTMs are useful because standard RNNs struggle with long-range dependencies and LSTMs help resolve this by introducing a memory cell and gates that store, update, and forget information over long time spans. Each LSTM unit has three gates (forget gate, input gate, and output gate) such that the forget gate decides what information should be removed from memory, the input gate decides what new information should be added, and the output gate decides what information should be sent out to the next step.

All three of these gates utilize sigmoids with values from 0 to 1 in order to control infomration flow, allowing it to remember information for long periods. Additionally, LSTMs are often used for sequence related tasks such as sequence classification and natural language processing (NLP), specifically text generation, machine translation, and sentiment analysis [9].

The basic operation in LSTMs is the following:

$$Wx + Uh + b \tag{8}$$

where x is the current input vector, h is the previous hidden state, W and U are weight matrices, and b is the bias. Additionally, at each time step t, the different gates have the following equations:

Forget gate (decides what to erase):

$$f_t = \sigma\left(W_f x_t + U_f h_{\{t-1\}} + b_f\right) \tag{9}$$

Input gate (decides what new information to store):

$$i_t = \sigma\left(W_i x_t + U_i h_{\{t-1\}} + b_i\right) \tag{10}$$

Output gate (decides what to output):

$$o_t = \sigma\left(W_o x_t + U_o h_{\{t-1\}} + b_o\right) \tag{11}$$

The model also relies on the sigmoid and relu functions for gates to decide how much to allow through:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{12}$$

As for our model training and evaluation using an LSTM, the input layer accepts a string of raw text, the text vectorization layer tokenizes and pads the strings, and the embedding layer converts word IDs to dense vectors. Additionally, the LSTM layer processes the sequence through bidirection and dropout and the dense output layers utilize relu activation and sigmoid activation functions. We then compiled the model using the adam optimizer, binary cross entropy for our loss function, and tracked accuracy for our metrics.

e) *Gradient Boosting:*

Gradient boosting is a supervised learning technique used primarily for regression and classification. It essentially builds a model such that each new model fixes the mistakes of the previous ones [10]. The general idea behind gradient boosting is that it builds a series of weak learners such as small decision trees. Each of these trees then focuses on the residuals made by the earlier trees and the final prediction is the sum of all the trees. The way gradient boosting works is that through an iterative process, you first start with a simple prediction, compute the errors using:

$$r_i = y_i - \hat{y_\imath} \tag{13}$$

train a small decision tree to predict these residuals, and add the tree to the model scaled by a learning rate:

$$\hat{y_\imath}^n = \hat{y_\imath}^o + \gamma * t(x_i) \tag{14}$$

In other words, gradient boosting fits each new tree to the negative gradient of the loss function.

As for our model training and evaluation using gradient boosting, we used XGBoost, which is an optimized machine learning algorithm that utilizes the gradient boosting framework. The way we implemented this model is that we first fit the XGBoost model using a logloss evaluation metric. Then, we predicted and evaluated on training data and utilized the classification report function to provide us with visual data such as precision, recall, f1-score, and support. Last, we predicted and evaluated on test data and also utilized the classification report function from the "sklearn.metrics" Python library. By doing so, we were able to easily compare the XGBoost training set performance against the XGBoost test set performance and find any differences and similarities between both performances.

f) *Multilayer Perceptrons (MLPs):*

Multilayer perceptrons are a feed-forward artificial neural network that only consists of dense layers with activation functions. In general, the model consists of an input, hidden, and output layers. Information from the previous layers are combined with their respective weights and move sequentially through the network to produce a binary output. Weights are updated trough a process called "back-propagation" in which gradient descent is applied to adjust weight values. [11]

MLPs are useful at capturing non-linear relationships and complex interactions between features which is helpful at analyzing text data due to the arbitrary vector values that are created when running lyrical data through the Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer, which convert text into numeric data. It is also relatively fast in computation compared to other neural networks such as CNNs and don't depend on the order of certain words which would be a problem if you are working with RNNs.

Our MLP model consists of converting the cleaned lyric dataset into a vector of floating point numbers, and running it through two dense hidden layers during training. The first hidden layer consists of 16 nodes with an Rectified Linear Unit (ReLU) activation function given by

$$f(x) = \max(0, x) \tag{15}$$

where x represents the input value. We then applied a dropout of 30% to avoid overfitting and had our second hidden layer consists of 8 nodes also with the ReLU activation function (15). In our case, we want to make sure that the model generalizes and finds patterns among lyrics part of the "adult" and "children" songs. Our output layer consists of one node with an activation function being the sigmoid function (4). This produces a value between 0 and 1, with our threshold being 0.5, allowing for binary classification of songs in our dataset. Our model was trained for 3 epochs with a batch size of 64, and had our validation set encompass 30% of our dataset.

III. Results

Our project undertook an evaluation of lyric classification models, testing Logistic Regression, Random Forest, SVM, XGBoost, Multilayer Perceptron

(MLP), LSTM across a meticulously curated dataset of 3,300 tracks with explicitness and age-appropriateness labels. Performance was assessed using precision, recall, and F1-score, ensuring that both the training and validation splits revealed each model's ability to generalize to new music content.

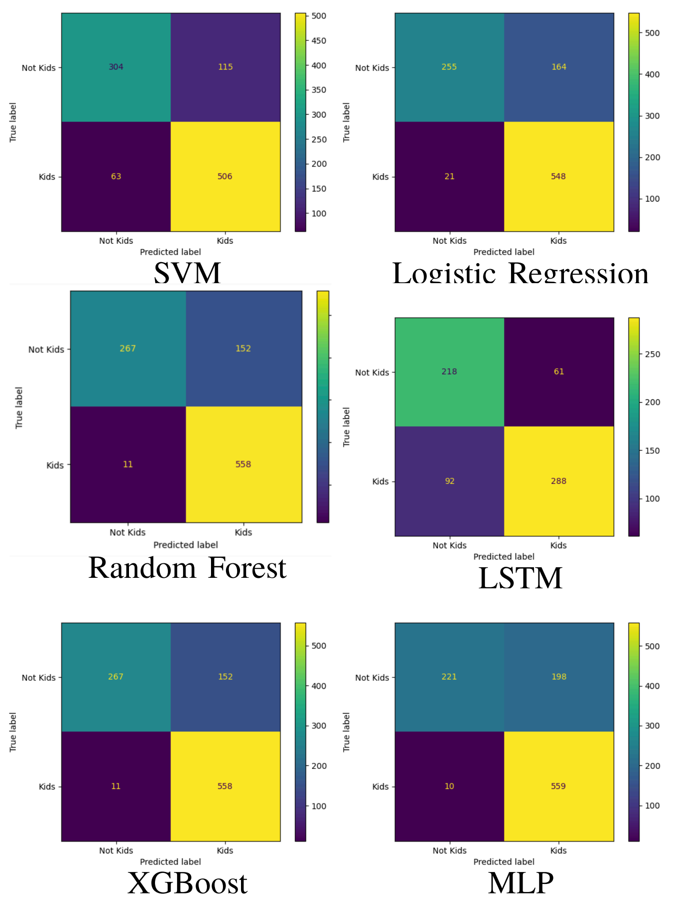


Fig. 2. Overall confusion matrices for all models

Among all methods, SVM and LSTM emerged as top performers: SVM excelled with a weighted training accuracy of 91%, while LSTM proved remarkably effective in handling complex, context-rich lyric sequences. The robustness of these models was especially evident when distinguishing tracks "safe for kids" from those containing explicit subtext. Although ensemble models (Random Forest, XGBoost) and deep learning approaches like MLP also delivered competitive accuracy, their recall and F1 scores were often limited when confronting subtle or ambiguous lyrical references. In parallel, Logistic Regression provided reliable performance for straightforward cases but struggled with more nuanced sequential relationships, underscoring the need for models that comprehend word order and context.

A critical insight from our experiments was the challenge of overfitting, particularly in neural models (LSTM, MLP) trained on a dataset of modest size. While these architectures achieved high training precision, validation scores, especially for minority or ambiguous categories, tended to trail, as highlighted by confusion matrix analyses (e.g. Fig. 2). The tendency to memorize unique phrase structures or keyword patterns became a limiting factor, especially with pronounced class imbalance and data scarcity. This is why, although LSTM has lower scores than many of our models, we still considered it by far one of the best. During our testing it was one of the least overfit models that we had. By contrast, SVM and Logistic Regression maintained stronger stability in test results and helped control overfitting, providing dependable baselines for content moderation workflows.

TABLE I
Weighted Average of Model performance comparison across metrics

	Precision	Recall	F-1 Score
SVM	0.82	0.82	0.82
Logistic Regression	0.84	0.81	0.80
Random Forest	0.86	0.84	0.83
LSTM	0.76	0.77	0.77
XGboost	0.86	0.84	0.83
MLP	0.83	0.79	0.77

Based on a balance of high accuracy, contextual sequence handling, and minimized overfitting, the LSTM models were selected for future deployment and enhancement. Their sequencing power allows for deep understanding of lyric flow and implicit meaning, placing them at the forefront of automated content moderation and large-scale playlist curation.

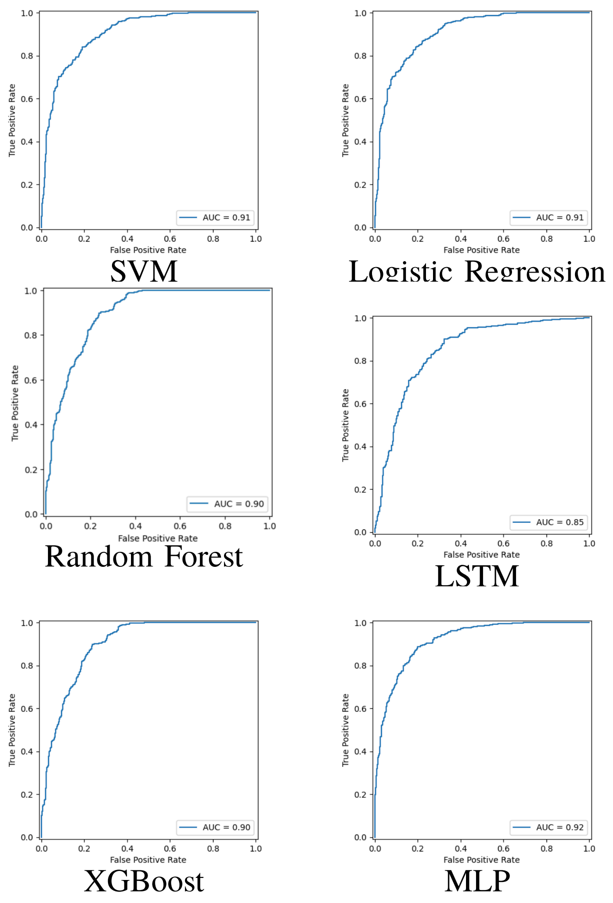


Fig. 3. Overall confusion matrices for all models

In conclusion, these results underscore the real-world promise of combining robust classical algorithms and advanced neural models for automated lyric screening. Deep learning models demonstrated crucial ability to uncover nuanced explicitness, and ongoing expansion of dataset size, plus integration of additional features like genre and sentiment, will enable even broader, more effective safeguards for family-friendly music platforms as catalogs continue to scale exponentially.

IV. Discussion

A. *Data Pipeline Considerations*

Through the comparative analysis of several models we tested, we were able to uncover performance characteristics based on their ability to handle the linguistic complexity of song lyrics. However, we also noticed that no single algorithm achieved optimal performance across all desired metrics: speed, generalization, and high recall. Each model we tested served different goals regarding content moderation, which we will further discuss below.

Throughout the process of building the dataset, we were able to use a custom-built dataset of 3,300 songs. Each row of data contained 4 attributes: artist, track name, lyrics, and age appropriate (our target variable). Next, we vectorized the song lyrics for each song as input for the machine learning model we wanted to test. Using spaCy, we tokenized the words, and we applied a technique called TF-IDF to convert to the significance of words in numerical data. The benefit of TF-IDF over other techniques such as Word2Vec was its higher accuracy for a more controlled dataset such as ours, as well as the automatic filtering of stop words and other common, non-relevant words for our use case.

B. *Model Trials*

After our data preprocessing, we focused on training each of the models based on what seemed to be most applicable from our research. We chose the following models: Logistic Regression, Random Forest, SVM, XGBoost, MLP, and LSTM. For our baseline model (Logistic Regression), we provided the TF-IDF vectors of the cleaned lyrics as input and fit them into the model with sklearn.

For the ensemble models (Random Forest, XGBoost), our main purpose was to maximize the detection of explicit content, which is essentially minimizing the risk of a false negative. Our Random Forest Classifier utilized 100 decision trees, 500 minimum samples to split a node, and 5 minimum samples in a leaf. Our numbers aligned with regularization techniques to reduce overfitting, as seen in our test and train precision, accuracy, and recall scores. XGBoost is also another model we tested to achieve a high recall for maximum detection of explicit songs. Our classifier used logistic loss (for binary classification), 50% of sampling for each tree before growing it, constructing 3 trees in parallel, with a learning rate of 0.1.

For the more advanced models, we used SVM, LSTM, and MLP. For SVM, we used LinearSVC to handle high dimensional TF-IDF with 80 iterations, and a regularization parameter of 0.1 (to mitigate overfitting). CalibratedClassifierCV was our wrapper around LinearSVC to determine probability for scoring metrics. Additionally, using a sigmoid function and 5-fold cross-validation, we trained our model. We initially defined an LSTM, however, upon noticing massive overfitting, we redirected our approach to Bidirectional LSTM to provide two LSTMS: one that runs forward and one that takes the input to process it backwards. This was an important differentiation, as taking both outputs allows us to analyze sentiments both before and after the word. Specifically for LSTM, we used TextVectorization for preprocessing (instead of spaCy like other models), as it provided a speed advantage for our deep learning model. The layers of LSTM were as follows: Preprocessing Layer (input, text vectorization, embedding), Sequential Processing (Bidirectional LSTM, dropout for regularization, L2 regularization, output regularization), and Output Layer (dense output and sigmoid activation function). We trained LSTM with the Adam optimizer and a learning rate of 0.1.

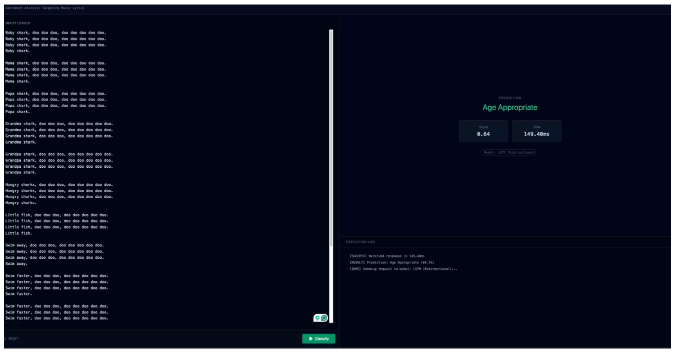
Lastly, we tested our data through MLP, multi-layer perceptron. This was a deeper version of logistic regression, used to handle the non linearities of the high dimensional TF-IDF vectors. Our model consists of the input layer, dense layer 1 (hidden layer that projects the high dimensional TF-IDF vector to 16 neurons using relu activation), dropout layer (to prevent overfitting), dense layer 2 (hidden layer to refine features to 8 neurons), and the output layer (sigmoid function). The importance of the sigmoid as our output activation function is necessary for us to determine the probability of an age-appropriate song (1) or not (0).

C. *Model Issues*

Upon training and testing the models, we ran into a few issues, the biggest one being overfitting. Across all the high-performance models, there is a gap between the training and testing accuracy. This big drop in accuracy indicates overfitting, and a root cause of this could be due to the small dataset size. Initially, we started off at 2700 songs, but due to severe overfitting issues, we increased the size of our data to 3300 songs. While this was able to reduce the gap of error between our training and testing accuracy, we still do have error between the two.

D. *Applications of the end Model*

We wanted our model to have real time applications to potential users who are interested. We did this by designing, and implementing a front end User Interface on top of our model, and hosting on Vercel. The user is able to launch the website, and access our already processed dataset of songs in near instant access. If a song is not currently in our database, the user can copy-paste lyrics, or any collection of text into our program to determine whether it is appropriate for children. See the UI elements below.



Baby shark, doo doo doo, doo doo doo doo doo.
Baby shark, doo doo doo, doo doo doo doo doo.
Baby shark, doo doo doo, doo doo doo doo doo.
Baby shark.

Mama shark, doo doo doo, doo doo doo doo doo.
Mama shark, doo doo doo, doo doo doo doo doo.
Mama shark, doo doo doo, doo doo doo doo doo.
Mama shark.

Papa shark, doo doo doo, doo doo doo doo doo.
Papa shark, doo doo doo, doo doo doo doo doo.
Papa shark, doo doo doo, doo doo doo doo doo.
Papa shark.

Grandma shark, doo doo doo, doo doo doo doo doo.
Grandma shark, doo doo doo, doo doo doo doo doo.
Grandma shark, doo doo doo, doo doo doo doo doo.
Grandma shark.

Grandpa shark, doo doo doo, doo doo doo doo doo.
Grandpa shark, doo doo doo, doo doo doo doo doo.
Grandpa shark, doo doo doo, doo doo doo doo doo.
Grandpa shark.

Hungry sharks, doo doo doo, doo doo doo doo doo.
Hungry sharks, doo doo doo, doo doo doo doo doo.
Hungry sharks, doo doo doo, doo doo doo doo doo.
Hungry sharks.

Little fish, doo doo doo, doo doo doo doo doo.
Little fish, doo doo doo, doo doo doo doo doo.
Little fish, doo doo doo, doo doo doo doo doo.
Little fish.

Swim away, doo doo doo, doo doo doo doo doo.
Swim away, doo doo doo, doo doo doo doo doo.
Swim away, doo doo doo, doo doo doo doo doo.
Swim away.

Swim faster, doo doo doo, doo doo doo doo doo.
Swim faster, doo doo doo, doo doo doo doo doo.
Swim faster, doo doo doo, doo doo doo doo doo.
Swim faster.

Swim faster, doo doo doo, doo doo doo doo doo.
Swim faster, doo doo doo, doo doo doo doo doo.
Swim faster, doo doo doo, doo doo doo doo doo.

```
[SUCCESS] Received response in 149.40ms
[RESULT] Prediction: Age Appropriate (64.1%)
[INFO] Sending request to model: LSTM (Bidirectional)...
```

Fig. 4. User Interface of our lyric classification model

Why is this important? There is validity in dedicating time to creating a front end for this project. It ties back into our mission and objective when starting this project to begin with. We want to enable parents to determine if the music that their children consume is indeed safe for them. There is a lot of music in recent times that is not considered explicit due to language, but the subtext hiding directly under the surface means that parents can miss very real instances where music sounds safe, but really isn't. A primary example of this is "Cake by the Ocean" by DNCE. This song is classified as "explicit" on Spotify, and other major streaming platforms, however they do have non-explicit versions that are available for children to listen to. The only reason that Spotify classified it as explicit initially is due to profanity. However the main sentiment of the lyrics is not about consuming cake. The issue lies wherein major music streaming platforms do not take the time to actually screen or monitor their music, to discern the subtext. There are many songs and "clean" or "non-explicit" versions of songs that still have highly graphic and not safe premises for children to be listening to.

Our model, using LSTM we were able to develop a proof of concept to truly discern subtext in songs, which is not an area deeply explored by major music providers. The mission was to create an interactive prototype where users could determine if text is safe for kids, which we were able to do successfully.

Throughout the front end design process, we had numerous considerations to take in. Did we want to allow users to add songs to the dataset? Did we want to set up API fetching in the front-end for lyric generation instead of copy pasting text directly? Our main factors were granular control, and scope. We wanted to enable our users to find and insert any lyrics they wanted, and preprocess them behind the scenes, instead of forcing them to make API calls. There was also the potential for the songs they would search to not be found in our APIs which we wanted to avoid. We also decided against the inclusion of new queries to the dataset, as that would cause our ratio of Adult to Kid friendly songs to potentially skew sharply in one direction. We decided to keep our interface simpler, and more in the hands of the user, which worked better for our timeline as well.

V. Conclusion

Our purpose and mission in this project was to create a better means to identifying graphic and explicit music, enabling parents to prevent exposure to graphic language, and concepts, via Machine Learning Methodologies. We were able to achieve this by meticulously combing through Spotify playlists, identifying and labeling thousands of songs as explicit or safe for children, and training numerous NLP models. Overall, this report showcases our journey from our original plan of unsupervised learning, the pivot after we realized a critical API we were planning on using had been deprecated, and our end adaption of training a supervised model to accurately determine if a song is explicit songs based on lyrics. Our current pipeline is first to query our "song grabber" script using a list of arbitrarily decided playlists generated by the Spotify community. These playlist have been assembled collectively, popularized and promoted to due their "safe for kids" and "explicit" songs.

Note: For accuracy purposes, we identified playlists and songs that are indeed highly explicit and may contain offensive language, for the purpose of providing meaningful context to our model. Please review the playlist names and song lyrics with this in mind.

After we grab the song names from Spotify, we fetch the lyrics from Genius's lyric API, creating separate datasets classified as "explicit" and "safe for kids", then merge them together, labeled accordingly. We also remove special characters as they are not considered relevant, due to them being using in positive and inappropriate settings unconditionally. The 3300 song dataset allows for accurate use of NLP models.

We then load the dataset into a data frame, clean and lemmatize the the lyrics using the SpaCy library. This is for noise removal purposes, to improve overall model performance. We then use feature engineering methodologies by converting the now cleaned lyrics to TF-IDF vectors to prep them for training. After various different models and algorithms, determined that the most accurate models were the SVM and the LSTM models. this is because both of them were able to glean the nuances in the sentiment of the lyrics, while avoiding memorizing the training data (over-fitting).

We ended up selecting the LSTM model. While there are difficulties with this model due to the high data ingestion requirements for training, this model is able to understand slang, and uncover the context of explicit song ideas, even when the words themselves might be "clean". This is unlike SVM, which trains faster, but will miss out on some nuances and explicit subtext.

With the technical difficulties, scope alignment, and working to ensure we have a complete MVP by the project deadline, we were not able to branch deeper into further refinding this project with nuance. There are multiple areas we would rework given more time. First, our dataset creation process is not at all scalable. Our pipeline ensures that we must at least somewhat screen the songs to ensure they are explicit/not explicit, when selecting. This won't scale if we want to provide larger datasets for our model, selection process has too much human involvement.

A proposed future feature would be to automate and create a consistent pipeline for dataset creation, to make it less biased, and have it solely pull from based on model decisions, compared to our random screening and selection. We would also benefit from modifying our User Interface, to use fuzzy matching in an attempt to fetch songs from an API, instead of having the user enter them into our front end manually. Lastly, more time would allow us to try to segment the classification scheme of songs further. Maybe instead of just having an "explicit", "safe for kids" tags, we could expand with "somewhat explicit" and "neutral" segments. A model with more data, and more time to hone in on the semantics of the lyrics would enable us to whittle down and be more specific. Overall, our model put more emphasis towards preventing children from hearing explicit music than the majority of music streaming providers(Spotify, Apple Music), by creating a systematic approach to lyric classification.

Overall, we believe that we contributed meaningful advances in the detection and analysis of music lyrics, in the context of screening and enabling parents to prevent their children from listening to explicit music. We experimented with numerous constraints, hyper parameters, and numerous supervised models to determine what could not only provide a meaningful proof of concept, but also what could scale. Our goal is that with the experiment, we can showcase that this is possible, and going forward with an LSTM model at scale can significantly improves the accuracy of those songs that are not for kids, and do not include profanity.

VI. Contributions

Adrian helped with creating the scripts to help generate the dataset and fine-tuning various models to avoid as much over-fitting as possible.

Maxim Helped write the abstract, discussion, and results, and fully wrote the conclusion. He also led the project management efforts, initial ideation, a, EDA graph visualization, and a large chunk of the slides.

Arezoo helped train and select the models, wrote the SVM slide, and made significant contributions to the results section of the paper.

Danny assisted in building the dataset using the lyric Genius API and Spotify API to query a significant number of songs and helped research various models for the team.

Divleen contributed to building and the selecting models and their architectures, along with a portion of the slides, and a majority of the discussion.

Aidan contributed by fully developing the frontend User Interface, testing, and recording the demo. He also helped with setting the the Github, and proper Github pruning practices.

Cory contributed with researching models, writing portions of the paper and slides, and integrating graphs onto the paper for clear result data.

Taylor contributed with data processing, cleaning and pipelining the training data, writing portions of the paper and slides, and finalizing the notebook.

References

[1] X. Zhang, "The Application of Natural Language Processing Technology Based on Deep Learning in Japanese Sentiment Analysis," in *2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE)*, 2023, pp. 1–5. doi: 10.1109/AIKIIE60097.2023.10390437.

[2] S. N., S. Wagle, P. Ghosh, and K. Kishore, "Sentiment Classification of English and Hindi Music Lyrics Using Supervised Machine Learning Algorithms," in *2022 2nd Asian Conference on Innovation in Technology (ASIANCON)*, 2022, pp. 1–6. doi: 10.1109/ASIANCON55314.2022.9908688.

[3] Z. Zhan, "Comparative Analysis of TF-IDF and Word2Vec in Sentiment Analysis: A Case of Food Reviews," *ITM Web of Conferences*, vol. 70, p. , 2025, doi: 10.1051/itmconf/20257002013.

[4] T. Evgeniou and M. Pontil, "Support Vector Machines: Theory and Applications," 2001, pp. 249–257. doi: 10.1007/3-540-44673-7_12.

[5] C.-P. Lee and C.-J. Lin, "A Study on L2-Loss (Squared Hinge-Loss) Multiclass SVM," *Neural Computation*, vol. 25, no. 5, pp. 1302–1323, May 2013, doi: 10.1162/neco_a_00434.

[6] N. R, B. R, R. Balamanigandan, and A. Priscilla, "Analyzing the Performance of Novel Logistic Regression over Linear Regression Algorithms for Predicting Fake Job with Improved Accuracy," in *2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2024, pp. 1728–1732. doi: 10.1109/ICESC60852.2024.10690033.

[7] R. Bollapragada, J. Nocedal, D. Mudigere, H.-J. Shi, and P. T. P. Tang, "A Progressive Batching L-BFGS Method for Machine Learning," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., in Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 620–629. [Online]. Available: https://proceedings.mlr.press/v80/bollapragada18a.html

[8] Y. Gu, "A Comparative Analysis Study of Stock Prediction Based on Random Forest and Decision Tree," in *2024 International Conference on Electronics and Devices, Computational Science (ICEDCS)*, 2024, pp. 96–100. doi: 10.1109/ICEDCS64328.2024.00022.

[9] R. K. Behera, M. Jena, S. K. Rath, and S. Misra, "Co-LSTM: Convolutional LSTM Model for Sentiment Analysis in Social Big Data," *Information Processing & Management*, vol. 58, no. 1, p. 102435, 2021, doi: 10.1016/j.ipm.2020.102435.

[10] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms." [Online]. Available: https://doi.org/10.1007/s10462-020-09896-5

[11] A. Oguntimilehin *et al.*, "A Multilayer Perceptron Based Mobile Diagnosis System for Malaria Fever," in *2024 International Conference on Science, Engineering and Business for Driving Sustainable

Development Goals (SEB4SDG), 2024, pp. 1–8. doi: 10.1109/SEB4SDG60871.2024.10629733.