MUSIC MOOD DETECTION

IIT Bombay
Arpana Prajapati
Gargi Bakshi
Palle Bhavana
Sathwika Reddy

Abstract

We explore different strategies to predict the mood of a song based on a combination of textual and non-textual features. The song's mood is classified as Happy, Sad, Calm, or Energetic. The performance of various models such as logistic regression, decision trees, ensemble classifiers, and neural networks is compared to select the one most suitable for this task. XGBoost classifier works best for the dataset used, giving a test accuracy of 82%. GitHub link to the project

1. Introduction

Music is far more powerful than language to stimulate strong emotions in humans. The correlation between music and emotion has always remained an interesting research domain in Music Psychology. Listening to music is a prominent component of human lives now, surpassing watching television and reading. Automatic Music Information Retrieval (MIR), which detects the emotion or mood dimensions of music is a growing research field. Previously, music classification tasks have primarily relied on features such as the track's audio signal or metadata. Lyrics features are also an important attribute along with audio features for song emotion classifications.

A project that uses the audio features of a song present as part of the song's metadata on Spotify [5] and a paper that detects the song's emotion based on its lyrics [2]. formed the basis of our project. The main goal of this project is to explore the mood detection of songs using both textual and audio features and check whether multimodal systems classify emotions in music better than singular models.

2. Dataset Development

Our final dataset contains 900 entries, of which about 200 were generated by us. The dataset has 18 features that belong to primarily two categories:

- **Song metadata**: song name, album, artist, ID, release date, popularity
- **Audio features**: length, danceability, acousticness, energy, instrumentalness, liveness, valence, loudness, speechiness, tempo, key, time signature

We have only one mood label associated with each song, and we later generate the lyrics to add to the dataset.

Table 1: *Mood Distribution in Dataset*

Mood	Count
Happy	205
Sad	229
Calm	229
Energetic	237
Total	**900**

2.1. Initial Dataset

We used the dataset available in the Spotify project's GitHub repository as our foundation, subsequently augmenting it with additional data. These audio features obtained from the Spotify audio analysis API are real numbers that measure how acoustic, positive, and intense/instrumental a track is based on rhythm stability, beat strength, and volume of the song. The audio features under consideration are indicative of extracted characteristics from an audio track. In the hypothetical scenario of manual feature extraction from the audio, it is plausible that metrics such as acousticness, positivity, and intensity/instrumentality, determined by factors like rhythm stability, beat strength, and volume, would have been obtained. The utilization of the spotipy API spared us from the manual extraction process.

2.2. Adding to the Dataset

To increase the size of the labelled dataset, we created playlists corresponding to each of the four moods using the MOOD playlist website [3]. The website creates playlists corresponding to moods energetic, mellow, upbeat, and downbeat. These moods relate to our labels, with happy corresponding to upbeat, sad corresponding to downbeat, and mellow corresponding to calm.

These playlists were used to obtain the Spotify data of the song using the spotipy library, in the same way as our reference project [5]. The dataset was then expanded by incorporating these entries.

2.3. Adding the Lyrics

The lyrics for the entire dataset were added using the lyricsgenius library [4]. Fig 1. shows the wordcloud for Sad lyrics. Word Clouds are visual representations of words that appear more frequently in the corpus.

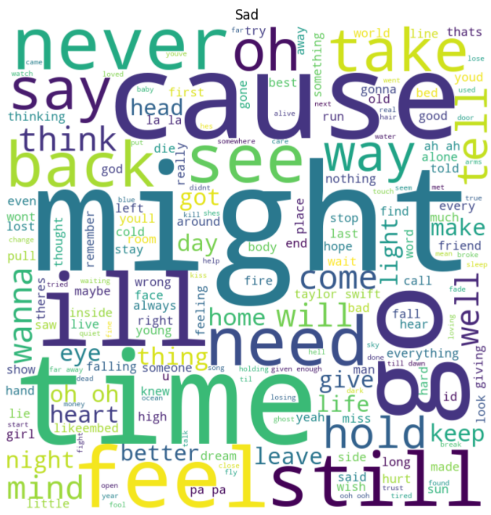


Figure 1: Wordcloud for sad song's lyrics

The lyrics were later preprocessed and tokenized so that they could be used as features. The details of this process are given below.

3. Preprocessing and Feature Extraction

This step involves the careful preparation and transformation of raw data into a format suitable for analysis, as well as the extraction of meaningful features from the lyrics that capture essential aspects of the underlying information.

3.1. Preprocessing

Some songs didn't have lyrics available through the lyricsgenius API, So we replaced these missing values with the name of the song. Some songs had extremely large text for lyrics which indicated that those lyrics weren't reliable, so we replaced all such lyrics with the song title. The langid library was used to detect the language of the lyrics to single out the entries which had non-English lyrics and in such cases, we again used the song name instead of the lyrics. After that, we removed all the non-alphabetical characters from the lyrics.

Preprocessing is an important step because the lyrics obtained were very noisy and some entries had non-lyrical text. Employing language and text-length checks helped reduce the number of such instances significantly, and we later observed that the performance of all models improved a lot due to this step.

3.2. Feature Extraction

For training the models, we needed both audio features and textual features from the lyrics. The necessary audio features are already present in the dataset. For extracting features from the lyrics, we used the Term Frequency and Inverse Document Frequency (TF-IDF) vectorizer to transform the pre-processed lyrics into a feature space consisting of 250 features.

TF-IDF is a numerical statistic obtained by multiplying two values - TF and IDF, where Term Frequency (TF) is the ratio of the frequency of the specific term in a document to the total number of terms in that document and Inverse Document Frequency (IDF) is the logarithm of the ratio of the total number of documents to the number of documents containing the term. It highlights terms that are both frequent within a document and rare across the corpus, emphasizing their significance.

TF-IDF vectorizer performs unigram tokenization of lyrics, removes stopwords, estimates the TF-IDF score for each token, and then selects the top 250 of them to act as the extracted features. We tuned the parameters of the TF-IDF vectorizer and observed that unigram tokens with 250 maximum features worked best for our dataset. Using Part-Of-Speech tagging as the tokenizer in TF-IDF (that is, replacing the default tokenizer with a process that assigns POS tags to each word and then using these tagged words as the terms for the TF-IDF matrix) did not improve the performance so we kept the default one for our model.

We observed that using lemmatization to reduce all the words in the lyrics to their root form did not yield much improvement in the utility of the textual features. We also checked if using nltk's sentiment intensity analyzer on the lyrics and adding the polarity scores as features improved the models' performance but in general, it resulted in a slight decrease in accuracy.

Extracting the most important features from the lyrics, and using those along with other non-textual features allowed the models to learn the nuanced relationship between a song and its underlying mood with greater accuracy.

4. Model Selection

We trained several models to see which one works best for our task.

4.1. Simple Classifiers

We trained models of logistic regression, a multi-layer perception, a decision tree, and a support vector classifier. All these worked fairly well, giving a test accuracy of around 70%.

4.2. Ensemble Classifiers

We compared the performance of Random Forest, XGBoost, and CatBoost models.

XGBoost: It stands for Extreme Gradient Boosting and is an implementation of gradient-boosting decision trees. It works by sequentially adding decision trees to correct errors made by the existing ensemble. It provides parallel tree boosting, where multiple weak models are combined to create a strong predictive model. XGBoost uses a user-defined objective function and regularization techniques to optimize model performance while preventing overfitting. Its strengths lie in efficient parallel processing and the ability to handle large datasets while delivering high prediction accuracy.
Tuning the number of estimators (=60) to optimize the performance of the model on our dataset gave us the highest test accuracy of 82%.

CatBoost: It stands for Categorical Boosting. Its gradient-boosting algorithm is similar to that of XGBoost. CatBoost stands out by having built-in support for categorical features and generally well-tuned default hyperparameters. Since our dataset has only numerical features, CatBoost did not improve the performance much, with a final test accuracy of 79%.

4.3. Neural Networks

We trained feed-forward neural networks with 2, 3, and 4 linear layers and ReLU activation. Their performance was similar to the performance of the statistical models.

We additionally implemented a neural network with group normalization (in which the activations are normalized based on groups rather than entire batches, which is the case with batch normalization). The aim of this was to reduce overfitting, which seemed to be happening with the unregularized models.

We also trained a CNN with a convolution layer, pooling, and two linear layers. This model did not perform that well due to the absence of visual features.

4.4. Attempts

In addition to the standard models, we attempted to construct a model whose architecture was detailed in a research paper on music mood recognition [1], which uses an autoencoder-based optimized support vector regression model. However, we obtained low test accuracies with this model, possibly due to errors in implementation and differences in the feature spaces and datasets.

5. Results

Best Model: The best model for our dataset was XGBoost, which gives a test accuracy of 82%.

Improvement by considering both audio and textual features: The best accuracy for models that consider only audio or only textual features is less than the best model obtained by considering both.

Table 2: *Best Accuracies*

Features	**Best Accuracy**
Textual	0.51
Audio	0.75
Both	**0.82**

This shows that using both lyrics and audio features helps in predicting the song's mood more correctly.

6. Analysis

Here are the salient observations about the various factors that affect the performance of our models:

6.1. Feature Extraction from Lyrics

The use of unigram textual features instead of bigrams or trigrams or their combinations gave better accuracy which suggests that the relationship between consecutive words might not be very influential for our specific mood detection task.

Figure 2: *Test Accuracies of different models*

Model	**Hyperparameters**	**Accuracy**
Logistic Regression	lbfgs solver	0.755
Multi-Layer Perceptron	hidden layer sizes: (8, 8, 8), max iter: 1500	0.724
Decision Tree	max depth: 5	0.782
Linear SVC	default	0.742
Random Forest	max depth: 5, 10 estimators	0.737
XGBoost	**60 estimators**	**0.822**
CatBoost	iterations: 1000, depth: 6, learning rate: 0.05	0.79
NN: 2 linear layers	hidden layer size: 300, learning rate: 0.01, 50 epochs	0.71
NN: 3 linear layers	hidden layer sizes: 128, 256; learning rate: 0.005, 70 epochs	0.76
NN: 4 linear layers	hidden layer sizes: 256; learning rate: 0.0001, 120 epochs	0.78
CNN	learning rate: 0.005, 80 epochs	0.70
Regularized NN	group sizes: 4, 8; hidden layer sizes: 128, 256, 256	0.79

Table 3: *Accuracy of SVC with different textual features*

n-gram combination	**Accuracy**
Uni	0.73
Uni + POS	**0.74**
Bi	0.71
Bi + POS	0.72
Tri	0.72
Tri + POS	0.70
Uni + Bi	0.73
Uni + Bi + POS	0.72
Bi + Tri	0.73
Bi + Tri + POS	0.74
Uni + Bi + Tri	0.73
Uni + Bi + Tri + POS	0.73

Adding sentiment intensity to the features did not improve performance, indicating that the emotional tone, as captured by sentiment analysis, may not be directly aligned with our mood categories. This might be due to the contextual ambiguity associated with lyrics, and the size and quality of the dataset.

Despite increasing processing time, switching to the BERT tokenizer did not result in an improvement in the test accuracy. This suggests that the complexity introduced by the BERT tokenizer did not offer any significant advantage for our specific task.

6.2. Simple Classifiers

Simpler models like Support Vector Classifiers (SVC), decision trees, and logistic regression outperformed unregularized neural networks in terms of accuracy. This could be because complex neural network models tend to memorize the data, and in the absence of regularization, these neural networks are more prone to overfitting. We can also note that overfitting is likely for this dataset because multiple labels may be possible for a song.

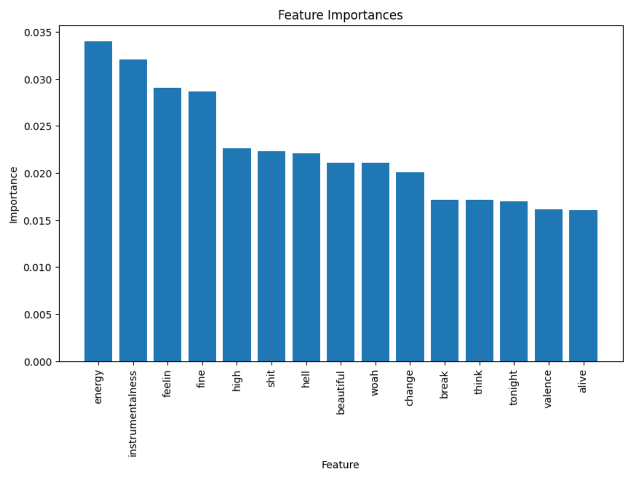


Figure 3: *Top 15 important features as learned by the XGBoost model*

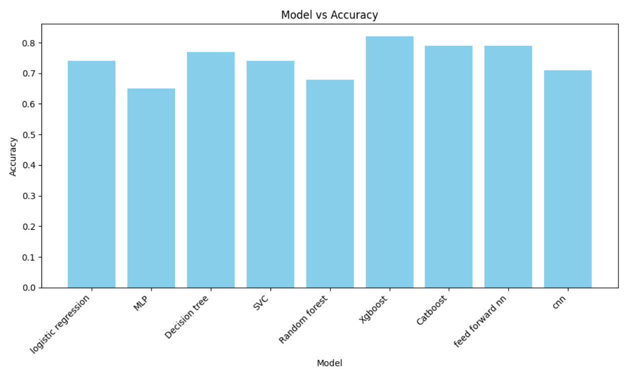
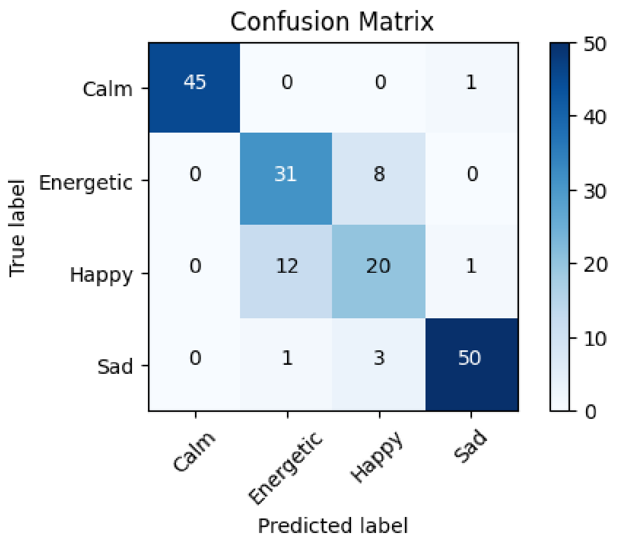


Figure 4: *Models with their test accuracies*

The addition of more data points to make the dataset balanced improved the generalization of our models. This adjustment resulted in more robust and reliable predictions, highlighting the importance of a balanced representation of different mood categories.

In assessing the performance of our models across different mood categories, the class labeled "Sad" consistently exhibited the highest correct predictions. Analysis of confusion matrices showed that a significant number of songs labeled "Happy" were misclassified as "Energetic" songs. This suggests a potential overlap or ambiguity between the characteristics of these two mood categories.

6.3. *XGBoost*



	Calm	Energetic	Happy	Sad
Calm	45	0	0	1
Energetic	0	31	8	0
Happy	0	12	20	1
Sad	0	1	3	50

Figure 5: *Xgboost confusion matrix*

The XGBoost model identified specific words such as "feeling", "fine", "woah", "high", and "hell" as highly influential in making predictions.

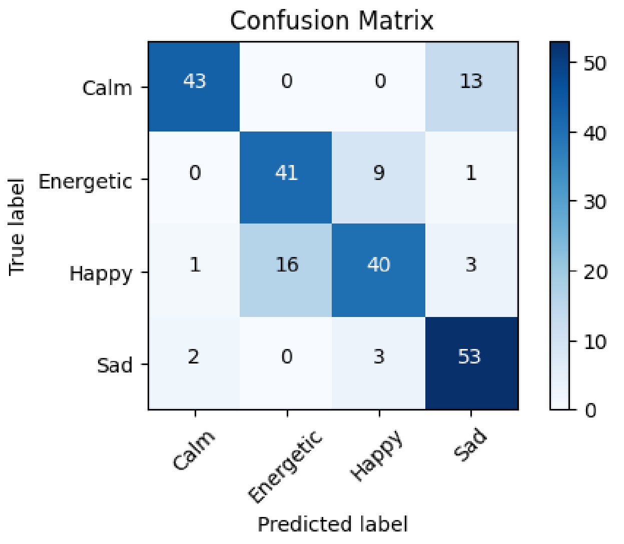
6.4. *Neural Networks*

Impact of hidden layers: The performance of neural networks showed a slight increase with the addition of more hidden layers. However, neural networks with 3 and 4 layers performed almost identically, suggesting a potential saturation point in terms of layer complexity.

CNN performance: The CNN did not outperform the feed-forward neural networks. This observation may be attributed to factors such as dataset size and the complexity of the mood detection task.

Effect of Group Normalization: The introduction of group normalization positively impacted model performance, improving the accuracy by 3%, suggesting that the unregularized neural networks were overfitting on the training data.

Accuracy disparities across mood categories: Calm songs exhibited lower prediction accuracy compared to the ensemble and statistical models. However, there was an improvement in the accuracy of predictions for energetic and happy songs. Overall, the regularized neural network generalizes well across all four mood categories.



	Calm	Energetic	Happy	Sad
Calm	43	0	0	13
Energetic	0	41	9	1
Happy	1	16	40	3
Sad	2	0	3	53

Figure 6: *Regularized NN confusion matrix*

7. Conclusion

We analyzed and implemented different approaches to predict music moods. We observed the role of lyrics and audio features in determining the emotion of a song and worked out methods to best extract the information available in these features. We noted the performance for different models, with regularized neural networks giving 0.79 accuracy and XGBoost giving the highest accuracy of 0.82.

References

[1] Gaurav Agarwal and Hari Om. “Music Mood Recognition”. In: (2021). URL: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/sil2.12015.

[2] *Lyrics Emotion Detection*. URL: https://github.com/imdiptanu/lyrics-emotion-detection.

[3] *Playlist Generator*. URL: https://www.moodplayl.ist/.

[4] *Song Lyrics*. URL: https://medium.com/analytics-vidhya/getting-lyrics-of-songs-via-python-lyricsgenius-23e5dd5992e3.

[5] *Spotify-based Mood Detection*. URL: https://github.com/cristobalvch/Spotify-Machine-Learning/tree/master.