Abstract

This project investigates mood-based classification using Spotify user data to develop a recommendation system. By analyzing user mood patterns and song attributes, we aim to create a model that enhances Spotify's recommendation capabilities. Key findings, methodologies, and the recommendation system's performance are discussed.

1 Introduction

1.1 Background

With the rise of personalized media experiences, platforms like Spotify seek ways to align content with user moods. Mood-based classification enables personalized song recommendations that reflect user emotions.

1.2 Motivation

Understanding user moods based on listening habits can enhance engagement and provide a deeply personal music experience, supporting Spotify's goal of tailored recommendations.

1.3 Objectives

The objectives of this project include:

- Developing a classification model to categorize user moods.
- Evaluating the system's performance.
- Integrating mood-based data into a recommendation algorithm.

2 Literature Review

This literature review focuses on two main areas: emotion recognition from audio data and recommendation techniques, specifically collaborative filtering and content-based filtering.

2.1 Existing Research

The ability to recognize emotions in music has been a significant research focus, with various methodologies explored:

- **Chroma Spectrograms and Deep Learning:** Recent studies have employed deep learning models, particularly using chroma spectrogram images, to classify music emotions. For instance, a study utilized the AlexNet architecture to analyze visual features derived from audio signals, achieving an accuracy rate of 86.3% in emotion recognition across different music pieces.

- **Feature Extraction Techniques:** Traditional methods often relied on extracting acoustic features such as rhythm, tone, and harmony. However, the challenge remained in accurately linking these features to specific emotional responses. Newer approaches have shifted towards utilizing deep learning frameworks that can process both time and frequency domain information more effectively.
- **Hybrid Approaches:** Some researchers have proposed combining audio signal analysis with lyrical content to improve emotion detection. By integrating multiple data sources, these methods aim to enhance the accuracy of emotion recognition systems.

2.2 Methodologies Used

In the realm of music recommendation systems, two predominant techniques are employed: collaborative filtering and content-based filtering. Based on the scope of this project, content-based filtering has been used.

- **Feature Similarity Analysis:** Content-based filtering generates recommendations based on the attributes of songs (e.g., genre, tempo, instrumentation). This technique assesses the similarity between songs' features and a user's listening history to suggest new tracks that align with their preferences.

3 Dataset Description

3.1 Source

The dataset consists of 12,04,025 instances and 24 features. It was sourced from Kaggle (`https://www.kaggle.com/datasets/rodolfofigueroa/spotify-12m-songs`), covering track features, and the user data was sourced from Spotify's API.

3.2 Features

Key features include danceability, energy, valence, tempo, loudness, key, acousticness, and instrumentalness, which contribute to mood classification.

- **Danceability:** Measures how suitable a track is for dancing, based on a combination of tempo, rhythm stability, and beat strength.
- **Energy:** Represents the intensity and activity level of a track, with higher values indicating more energetic songs.
- **Valence:** Indicates the musical positivity of a track, with higher values representing more cheerful and uplifting tones.
- **Tempo:** The overall speed or pace of the track, measured in beats per minute (BPM).

- **Loudness:** Measures the average volume of the track in decibels (dB), normalized across all tracks.
- **Key:** The musical key of the track, represented as an integer corresponding to standard musical key notation.
- **Acousticness:** Predicts the likelihood that a track is acoustic, with higher values indicating stronger acoustic attributes.
- **Instrumentalness:** Estimates the degree to which a track is instrumental, with higher values implying fewer or no vocal elements.

3.3 Target Variable

The target variable is user mood, which was not present in the dataset has been defined seperately. The user moods have been classified into seven different categories such as calm, dancing, energetic, happy, neutral, relaxing and sad .

4 Data Preprocessing

4.1 Cleaning and Preparation

The dataset exhibited exceptional quality, with all feature values being well-structured and consistent, thereby requiring no additional cleaning.

To address varying feature scales, the `loudness` and `tempo` attributes, which had significantly higher ranges, were scaled using a Standard Scaler. This ensured these features contributed proportionately to the analysis without dominating due to their magnitude.

4.2 Handling Missing Values and Outliers

The dataset was thoroughly inspected for quality and consistency before analysis. Notably, there were no missing values across any features, ensuring the integrity of the dataset for subsequent processing.

To identify and remove outliers, a box plot-based approach was employed for the numeric features, including `loudness`, `danceability`, `energy`, `speechiness`, and others. Using the interquartile range (IQR) method, outliers were defined as values lying beyond 1.5 times the IQR from the first or third quartiles. These values were systematically filtered out to ensure a cleaner dataset. Box plots were then visualized for each feature to confirm the absence of extreme values and validate the refinement process.

4.3 Feature Selection and Engineering

The features used in the model were selected by experimenting with various combinations derived from Random Forest feature importance and scikit-learn's feature selection methods. After evaluating their performance, the most relevant features were retained. No additional feature engineering was deemed necessary as the existing features adequately captured the required information for the task at hand.

5 Methodology

5.1 Algorithms Used

This study utilized a variety of machine learning algorithms, ranging from classical models to advanced ensemble methods, to address the classification task effectively. To address the issue of class imbalance, the Synthetic Minority Oversampling Technique (SMOTE) was applied to the training dataset. This step ensured that minority classes were adequately represented, thereby improving model performance.

- **Logistic Regression (LR):** A linear classification model widely used for its simplicity and effectiveness in linearly separable datasets. It was also employed as the meta-classifier in the stacking ensemble.
- **Support Vector Classifier (SVC):** A robust algorithm that constructs hyperplanes to separate classes effectively, especially in high-dimensional spaces.
- **Stochastic Gradient Descent Classifier (SGD):** A linear classifier that uses iterative optimization techniques for scalable learning on large datasets.
- **Decision Tree (DT):** A non-parametric model that partitions the dataset into subsets based on feature conditions, offering interpretability and flexibility.
- **Random Forest (RF):** An ensemble method based on decision trees, which improves accuracy by averaging predictions from multiple trees.
- **AdaBoost:** An ensemble boosting algorithm that combines multiple weak classifiers, typically decision trees, to create a strong classifier by focusing on previously misclassified samples.
- **XGBoost (XGB):** A gradient boosting framework known for its computational efficiency and advanced regularization techniques.
- **LightGBM (LGB):** A histogram-based gradient boosting library designed for high speed and scalability, capable of handling large datasets efficiently.

- **Cosine Similarity for Song Recommendation:** For recommending songs, a similarity matrix based on cosine similarity between scaled audio features was computed. Each song in the new data was matched to the most similar songs from a large dataset, leveraging features such as acousticness, valence, loudness, energy, and tempo.

5.2 Justification of Model Choices

The choice of algorithms in this study was guided by the need to balance simplicity, interpretability, computational efficiency, and predictive performance. Each model served a specific purpose in addressing the challenges posed by the dataset and the classification task:

- **Logistic Regression, SVC, and SGD Classifier:** These linear models were included for their simplicity and ability to perform well on linearly separable data. Logistic Regression provided a baseline for comparison, while SVC and SGD offered alternative optimization approaches for classification.
- **Decision Tree:** The Decision Tree model was chosen for its interpretability and ability to capture non-linear patterns in the data. It also formed the foundation for more advanced ensemble methods.
- **Random Forest:** Random Forest mitigated the overfitting tendency of individual Decision Trees by averaging predictions over multiple trees. This model was well-suited for handling high-dimensional data and provided robust results.
- **AdaBoost:** AdaBoost focused on improving the accuracy of weak learners by iteratively adjusting weights for misclassified samples. It offered a complementary perspective to the bagging-based Random Forest.
- **XGBoost and LightGBM:** These gradient boosting algorithms were selected for their efficiency and high predictive power. Both XGBoost and LightGBM are capable of handling large datasets effectively, with LightGBM being particularly advantageous in terms of speed.
- **Ensemble Methods (Stacking and Soft Voting):** Ensemble methods were employed to leverage the complementary strengths of individual models. Stacking used a meta-classifier to combine predictions, while weighted soft voting aggregated predictions to enhance robustness and accuracy.
- **SMOTE:** The application of SMOTE ensured that the models were trained on balanced datasets, enhancing their ability to predict minority class labels accurately.
- **Cosine Similarity for Song Recommendation:** The use of cosine similarity enabled the recommendation of songs that closely matched the

audio feature profiles of new data. This method effectively personalized recommendations based on the mood predicted by the models.

This holistic approach ensured that the predictive models and recommendation system were both accurate and scalable, meeting the study's objectives effectively.

6 Implementation

6.1 Tools and Libraries

The following software and libraries were utilized for data analysis, model training, evaluation, and recommendation system development:

- **Programming Language:** Python was the primary language used for implementing all machine learning models and the recommendation system.
- **Data Manipulation and Preprocessing:**
  - `pandas`: For data manipulation and cleaning.
  - `numpy`: For numerical computations.
  - `scipy.stats`: For statistical operations, including z-score calculations.
  - `sklearn.preprocessing`: For data standardization (`StandardScaler`) and label encoding (`LabelEncoder`, `OrdinalEncoder`).
  - `imbalanced-learn (SMOTE)`: For handling class imbalance using Synthetic Minority Oversampling Technique.
- **Exploratory Data Analysis and Visualization:**
  - `matplotlib`: For creating plots and visualizations.
  - `seaborn`: For advanced data visualization and heatmaps.
- **Model Training and Evaluation:**
  - `scikit-learn`: For implementing classification models (Logistic Regression, SVC, SGDClassifier, DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier), preprocessing steps, and performance metrics.
  - `XGBoost` and `LightGBM`: For high-performance gradient boosting models.
  - `scikit-learn.metrics`: For performance evaluation, including classification reports, confusion matrices, and AUC-ROC curves.
- **Recommendation System:**

  - `sklearn.metrics.pairwise (cosine_similarity)`: For calculating song similarity based on audio features.

- **Pipeline Management and Utilities:**
  - `joblib`: For saving and loading trained models.
  - `sys` and `time`: For runtime calculations and system utilities.

6.2 Important Hyperparameters and Their Settings

- **Logistic Regression:**
  - `solver:` saga – Chosen for its ability to handle large datasets efficiently.
  - `max_iter:` 1000 – Ensures convergence for complex datasets.
  - `penalty:` l2 – Provides regularization to avoid overfitting.
- **SGD Classifier:**
  - `loss:` log_loss, hinge – Supports both logistic regression and SVM-style objectives for versatility.
  - `alpha:` 0.0001 – Provides optimal regularization for the dataset.
- **Decision Tree Classifier:**
  - `max_depth:` 10 – Limits the tree depth to prevent overfitting.
  - `min_samples_split:` 10 – Ensures splits are performed only with sufficient data, improving generalization.
- **Random Forest Classifier:**
  - `n_estimators:` 100 – Balances performance and computation time.
  - `max_depth:` 10 – Ensures trees do not grow excessively deep, improving efficiency and avoiding overfitting.
- **AdaBoost Classifier:**
  - `estimator:` DecisionTreeClassifier with `max_depth:` 10 and `class_weight:` balanced – Reduces overfitting and handles imbalanced datasets.
  - `n_estimators:` 200 – Provides sufficient iterations to build a strong ensemble.
  - `learning_rate:` 0.5 – Balances the weight of each estimator, avoiding excessive emphasis on individual classifiers.
- **LightGBM Classifier:**
  - `objective:` multiclass – Specifies the problem as a multiclass classification task.

- `num_class`: number of unique classes in the target – Adjusts the number of classes for multiclass classification.
- `boosting_type`: gbdt – Chooses gradient boosting decision trees for model training.
- `num_leaves`: 31 – Sets the number of leaves in each tree, balancing model complexity and performance.
- `max_depth`: 6 – Limits tree depth to reduce overfitting and improve generalization.
- `learning_rate`: 0.08 – Optimizes the learning process by reducing the step size during training.
- `feature_fraction`: 0.85, `bagging_fraction`: 0.85 – Balances overfitting and performance by selecting random subsets of features and data.
- `min_data_in_leaf`: 45 – Ensures that leaf nodes contain sufficient data to prevent overfitting.
- `lambda_l1`: 0.15, `lambda_l2`: 0.15 – Regularizes the model to avoid overfitting, helping improve generalization.
- `max_bin`: 255 – Sets the maximum number of bins for categorical features to improve accuracy.
- `min_gain_to_split`: 0.1 – Specifies the minimum gain for a split, ensuring only useful splits are made.

- **SMOTE (Synthetic Minority Over-sampling Technique):**
  - `sampling_strategy`: auto – Ensures that all classes are resampled to the majority class size, providing a balanced dataset for training.
  - `random_state`: 42 – Ensures reproducibility of results by fixing the random seed.
  - `k_neighbors`: 5 – Defines the number of nearest neighbors to use for generating synthetic samples.
  - `m_neighbors`: 10 – Specifies the maximum number of neighbors to use for each synthetic instance, which helps in controlling the oversampling behavior.

- **Stacking Ensemble:**
  - `estimators`: [('rf', RandomForestClassifier), ('xgb', XGBClassifier), ('lgb', LGBMClassifier)] – The base models used in the stacking ensemble.
  - `final_estimator`: LogisticRegression – The meta-model used to combine the predictions of the base models.
  - `cv`: 5 – Cross-validation strategy for training the base models.

- `passthrough:` True – If True, allows the final estimator to also use the original features in addition to the predictions of the base models.
- `stack_method:` auto – Defines the method used to generate the input to the meta-model (can be 'predict_proba' or 'predict').

- **Soft Voting Ensemble:**
  - `estimators:` [('rf', RandomForestClassifier), ('xgb', XGBClassifier), ('lgb', LGBMClassifier)] – The base models used in the soft voting ensemble.
  - `weights:` [0.35, 0.35, 0.3] – The weights assigned to each model based on performance.
  - `voting:` soft – Soft voting uses predicted probabilities rather than hard labels to combine model predictions.

6.3 Training Process

In the training process, a stratified sampling approach was employed to ensure that each class was represented proportionally in both the training and testing sets. The dataset was divided such that 80% of the data was allocated for training, while the remaining 20% was set aside for testing. The training data was then used to optimize the model parameters, while the test data provided an independent evaluation of the model's performance.

7 Results

7.1 Performance Evaluation

1. **Accuracy**:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Where: - TP = True Positives - TN = True Negatives - FP = False Positives - FN = False Negatives

2. **Precision**:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision measures the proportion of positive predictions that are actually correct.

3. **Recall (Sensitivity or True Positive Rate)**:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Recall measures the ability of the model to identify all positive samples.

4. **F1-Score**:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1-score is the harmonic mean of Precision and Recall, providing a balance between the two.

5. **Confusion Matrix**: The confusion matrix is a table used to evaluate the performance of a classification model, showing the number of correct and incorrect predictions for each class. It is defined as:

$$\begin{bmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{bmatrix}$$

Where: - TP = True Positives (correctly predicted positive cases) - TN = True Negatives (correctly predicted negative cases) - FP = False Positives (incorrectly predicted as positive) - FN = False Negatives (incorrectly predicted as negative)

7.2 Comparison of Models

Model	Accuracy	F1 Score	Precision	Recall
Logistic Regression	0.772	0.754	0.764	0.772
SGD Classifier	0.679	0.653	0.703	0.679
SVC	0.843	0.796	0.773	0.825
Decision Tree	0.874	0.874	0.875	0.874

Table 1: Performance Comparison of Various Models.

Model	Accuracy	F1 Score	Precision	Recall
Random Forest	0.920	0.918	0.917	0.919
AdaBoost	0.880	0.882	0.895	0.879
LightGBM	0.910	0.911	0.914	0.914
LightGBM (with SMOTE)	0.890	0.900	0.910	0.890
Stacking Ensemble	0.900	0.900	0.920	0.900
Soft Voting Ensemble	0.910	0.910	0.920	0.910

Table 2: Performance Comparison of Ensemble Learning Methods

Class	Precision	Recall	F1 Score	Support
Calm	0.97	1.00	0.98	10647
Dancing	0.74	0.78	0.76	5046
Energetic	1.00	0.97	0.99	24564
Happy	0.81	0.92	0.87	35996
Neutral	0.96	0.98	0.97	63059
Relaxing	1.00	0.96	0.98	30120
Sad	0.70	0.45	0.55	13911

Table 3: Performance of Classes in Random Forest Model

Class	Precision	Recall	F1-Score	Support
Calm	0.97	1.00	0.98	10647
Dancing	0.61	0.96	0.74	5046
Energetic	1.00	0.97	0.99	24564
Happy	0.87	0.78	0.82	35996
Neutral	0.97	0.95	0.96	63059
Relaxing	1.00	0.96	0.98	30120
Sad	0.55	0.69	0.61	13911

Table 4: Stacking Ensemble Performance

Class	Precision	Recall	F1-Score	Support
Calm	0.97	1.00	0.98	10647
Dancing	0.62	0.95	0.75	5046
Energetic	1.00	0.97	0.99	24564
Happy	0.84	0.86	0.85	35996
Neutral	0.97	0.95	0.96	63059
Relaxing	1.00	0.96	0.98	30120
Sad	0.61	0.58	0.60	13911

Table 5: Soft Voting Ensemble Performance

Class	Ensemble	RF	XGB	LGB
Calm	0.999	1.000	0.990	0.998
Dancing	0.953	0.966	0.768	0.974
Energetic	0.972	0.972	0.970	0.971
Happy	0.857	0.811	0.925	0.768
Neutral	0.953	0.949	0.977	0.943
Relaxing	0.961	0.962	0.958	0.960
Sad	0.582	0.625	0.439	0.696

Table 6: Performance of Different Classes for Soft Voting Classifier

7.3 Visualizations

The following visualizations provide insights into the performance of the classification models and highlight key metrics for comparison.

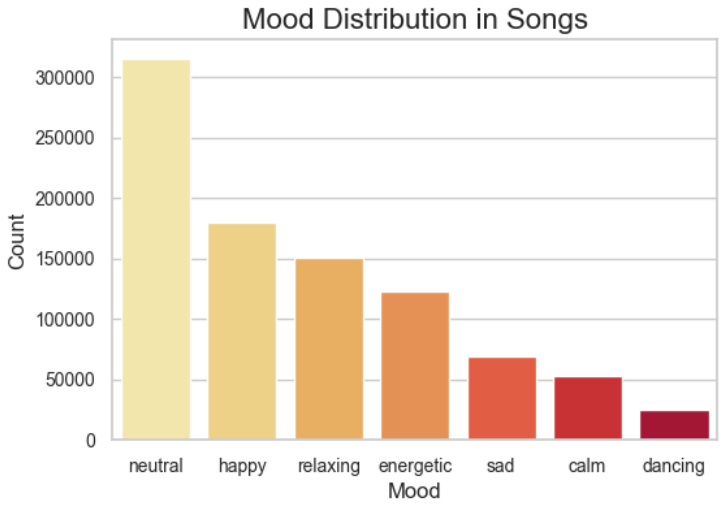

Figure 1: Mood Distribution in the Dataset

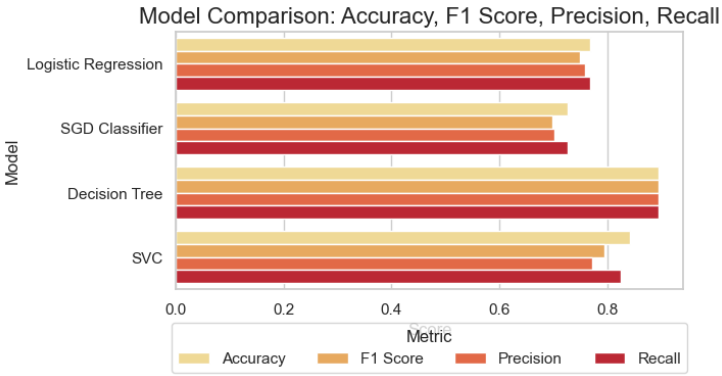

Figure 2: Classification Model Comparison.

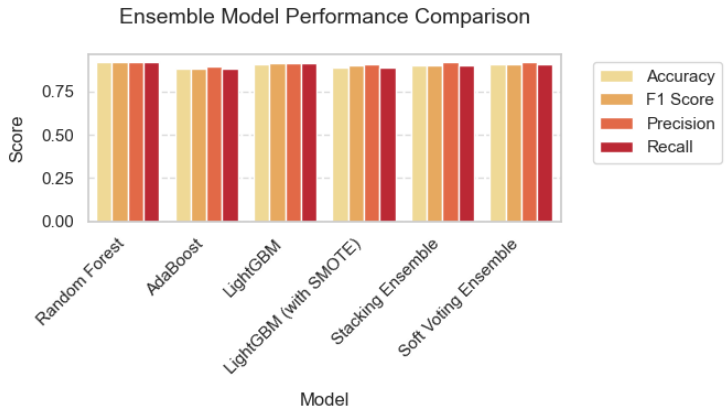

Figure 3: Performance Comparison for Ensemble Learning Models

Figure 4: Individual Class Performance for Random Forest Classifier

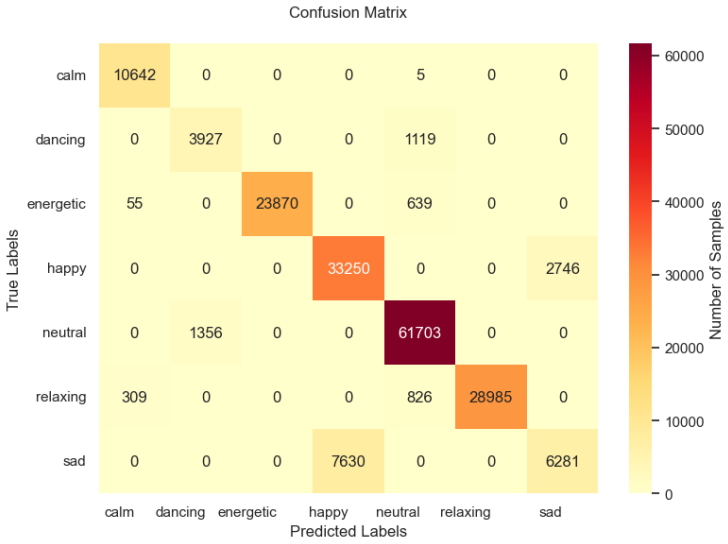

Figure 5: Confusion Matrix for Random Forest Classifier

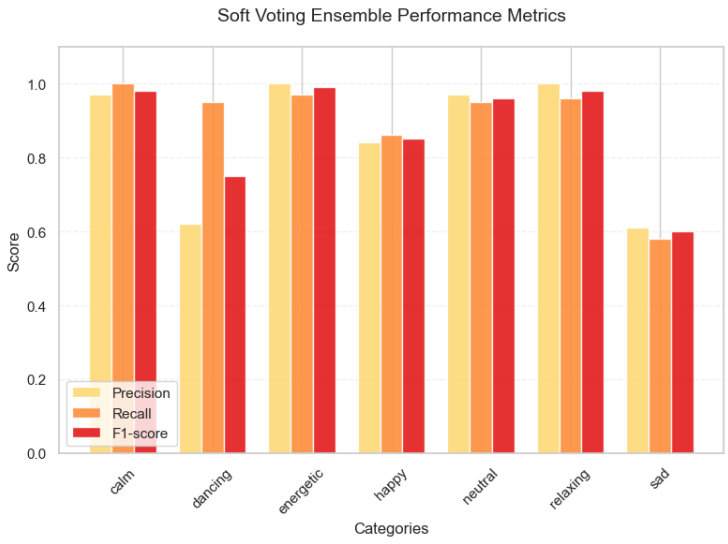

Figure 6: Individual Class Performance for Soft Voting Ensemble

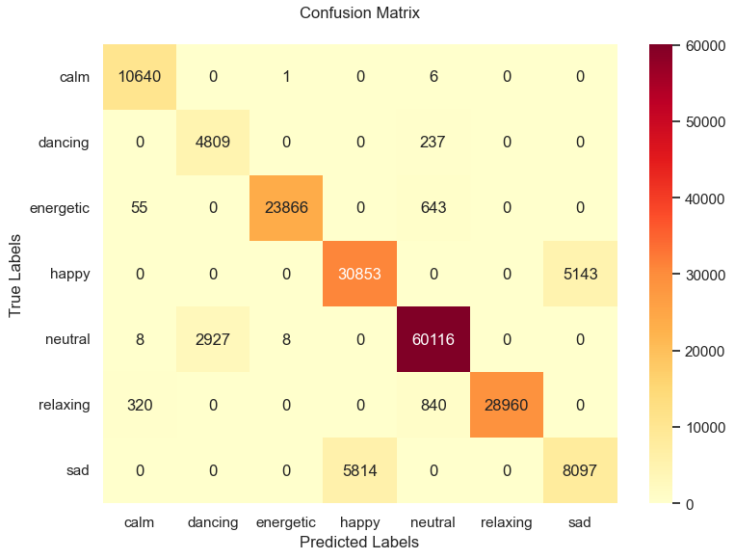

Figure 7: Confusion Matrix for Soft Voting Ensemble

8 Discussion

8.1 Interpretation of Results

Mood Distribution in the Dataset

The analysis of the large dataset reveals the following mood distribution:

- **Neutral:** 315,293 songs (most dominant mood, indicating a preference for balanced or emotionally neutral tracks in the dataset).
- **Happy:** 179,979 songs (second most common mood, showcasing an inclination for uplifting music).
- **Relaxing:** 150,600 songs.
- **Energetic:** 122,819 songs.
- **Sad:** 69,555 songs (less frequent but significant, highlighting emotional depth).
- **Calm:** 53,235 songs.
- **Dancing:** 25,232 songs (least represented mood, suggesting less emphasis on dance-oriented music in the dataset).

User Playlist Mood Distribution

The user's playlist, consisting of 50 songs, was classified into the following moods using the soft voting ensemble:

- **Sad:** 31 songs (62% of the playlist).
- **Energetic:** 19 songs (38% of the playlist).

Trends

- **Emotional Polarization:** The user's playlist is predominantly divided between two contrasting moods: **Sad** (introspective, emotional) and **Energetic** (lively, dynamic).
- **Lack of Variety:** No representation of other moods like **Neutral**, **Happy**, or **Relaxing**, which are more prevalent in the larger dataset.

Anomalies

The absence of **Neutral** mood songs in the user playlist is surprising, given its dominance in the larger dataset. This suggests that the user's listening preferences are focused on specific emotional states.

User-Centric Insights

The playlist's strong leaning toward **Sad** and **Energetic** moods reflects specific emotional and activity-driven preferences. This information can guide the recommendation system to prioritize these moods when suggesting new songs.

Recommendation Strategy

To enhance the listening experience, the following strategies can be adopted:

- Focus on recommending songs with similar audio features and mood profiles (**Sad** and **Energetic**) to align with current preferences.
- Introduce complementary moods like **Relaxing** (to balance **Sad**) or **Happy** (to enhance **Energetic**) for greater diversity.

Connection to the Large Dataset

The large dataset, with its diverse mood distribution, provides an extensive pool of songs for generating personalized recommendations. By leveraging audio features and cosine similarity, songs closely aligned with the user's playlist can be identified, ensuring both relevance and consistency.

8.2 Limitations

While the project achieved its primary objectives, several limitations were identified:

- **Absence of True Mood Labels:** The dataset used for this project did not contain true mood labels; instead, these labels were defined later using specific thresholds for audio features. Different threshold values or methods for defining moods could lead to varying results, introducing subjectivity in the labeling process.
- **Impact of Lyrics on Mood:** The recommendation system relies solely on audio features, which, while effective, cannot capture the emotional impact of song lyrics. Songs with similar audio features may evoke different moods depending on their lyrical content, which the current approach does not account for.
- **Model Performance on Underrepresented Classes:** The ensemble model struggled with underrepresented mood categories, such as *Dancing* and *Calm*, resulting in lower prediction accuracy for these moods. This could affect the diversity of the recommended playlists.
- **Cosine Similarity Assumptions:** The recommendation system is based on cosine similarity of audio features, which assumes these features fully encapsulate mood and user preferences. This approach may overlook contextual factors such as genre or cultural nuances.
- **Limited Playlist Size:** The user's playlist contained only 50 songs, which may not provide a comprehensive representation of their preferences. This could limit the recommendation system's ability to generalize effectively.
- **Class Imbalance in the Dataset:** The larger dataset exhibited a significant class imbalance, with moods like *Neutral* and *Happy* being overrepresented. Although class weighting was used to mitigate this, the imbalance may still bias the system toward these classes.
- **User Feedback Integration:** The recommendation system does not yet incorporate real-time user feedback, which could refine and personalize suggestions over time.
- **Dependency on Audio Features:** The system's reliance on numerical audio features, while efficient, may overlook other nuanced aspects of music, such as collaborative dynamics, emerging trends, or listener-specific sentimental associations.

Future work should address these limitations to enhance the robustness, personalization, and reliability of the system.

9 Conclusion

9.1 Summary of Findings

This project aimed to classify the moods of songs in a user's playlist, analyze their listening preferences, and develop a recommendation system based on audio features. The key outcomes and objectives achieved are summarized below:

Classification of Song Moods

- A soft voting ensemble combining **Random Forest**, **XGBoost**, and **LightGBM** models was developed, achieving robust mood predictions for songs based on audio features.
- The ensemble was evaluated against individual model performances, with weighted predictions ensuring optimal accuracy and generalization.
- The user's playlist (50 songs) was classified, revealing a significant inclination toward two moods:
  - **Sad:** 62% of songs.
  - **Energetic:** 38% of songs.

Recommendation System Development

- A recommendation system was designed based on **cosine similarity** of audio features, enabling personalized suggestions from the larger dataset.
- Recommendations prioritized **Sad** and **Energetic** moods to align with the user's preferences.
- The system introduced the potential for diversity by suggesting complementary moods such as **Relaxing** or **Happy**.

9.2 Objectives Met

1. Successfully classified song moods using a machine learning ensemble, meeting the goal of mood detection.
2. Extracted and analyzed mood trends in both the user's playlist and the large dataset, fulfilling the objective of understanding listening preferences.
3. Developed a recommendation engine leveraging audio features and similarity metrics, enabling personalized song recommendations.
4. Provided insights into mood trends and anomalies, facilitating a deeper understanding of the user's emotional engagement with music:

- A recommendation system was designed based on **cosine similarity** of audio features, enabling personalized suggestions from the larger dataset.
- Recommendations prioritized **Sad** and **Energetic** moods to align with the user's preferences.
- The mood distribution of the recommended playlist was:
  - **Sad:** 35 songs.
  - **Energetic:** 15 songs.
- The system introduced the potential for diversity by suggesting complementary moods such as **Relaxing** or **Happy**.

9.3 Future Work

Building upon the current work, several enhancements and expansions can be implemented to improve the system's performance, scalability, and user experience:

- **Web Application Development:** Implementing the recommendation system as a web application using frameworks like Flask. This would provide users with an interactive interface to upload their playlists, view mood classifications, and receive personalized recommendations.
- **Database Integration:** Incorporating a robust database system to handle larger datasets efficiently. Technologies such as PostgreSQL or MongoDB can facilitate faster querying and scalable storage for millions of songs and their associated features.
- **Lyrics and Audio Combined Labels:** Expanding the model to include lyrics as a feature by employing Natural Language Processing (NLP) techniques. Combining lyrical sentiment analysis with audio features could result in more accurate mood predictions, capturing both musical and textual aspects of songs.
- **Dynamic User Feedback Mechanism:** Integrating real-time feedback to continuously refine and personalize recommendations based on user interactions. Machine learning models can be fine-tuned with this feedback for improved accuracy.
- **Advanced Feature Engineering:** Exploring additional audio features or feature extraction techniques, such as deep learning-based embeddings, to capture complex musical patterns that may influence mood.
- **Scalability Improvements:** Implementing advanced indexing methods like Approximate Nearest Neighbors (ANN) to handle similarity calculations efficiently for large-scale datasets. Dimensionality reduction techniques like PCA or t-SNE could also be explored.

- **Mood Label Refinement:** Investigating more robust methods for defining mood labels, such as clustering techniques or crowdsourced annotations, to reduce subjectivity in the labeling process.
- **Collaborative Filtering Techniques:** Incorporating collaborative filtering approaches to supplement content-based recommendations, leveraging user similarity and past preferences to suggest songs.

These enhancements would make the system more robust, scalable, and personalized, addressing the identified limitations and meeting diverse user needs effectively.

10 References

References

[1] Milad Nooraei, *Spotify Analytics: An extensive music data analysis and recommendation project, encompassing data scraping, cleaning, predictive modeling, clustering, and interactive data visualization*, GitHub, `https://github.com/MiladNooraei/Spotify-Analytics`, 2021.

[2] Cristobal Valenzuela-Chavez, *Spotify Machine Learning*, GitHub, `https://github.com/cristobalvch/Spotify-Machine-Learning`, 2021.

[3] Shrunali Sali, *Spotify Data Visualization*, Medium, `https://medium.com/@shrunalisalian97/spotify-data-visualization-4c878c8114e`, 2021.

[4] Kaggle, *Spotify Recommender System*, Kaggle, `https://www.kaggle.com/datasets/geomack/spotify-recommender-system`, 2021.

[5] OpenAI, *ChatGPT*, OpenAI, `https://openai.com/chatgpt`, 2023.

Kaggle, *Kaggle Datasets*, `https://www.kaggle.com/datasets`, 2021.