

# Hit Vision - Predict the future of your song

1. Abstract
2. Introduction
3. Literature Review
  - 3.1 Feature Extraction in Music Analytics
  - 3.2 Predictive Modeling for Song Success
  - 3.3 Challenges and Gaps in Existing Research
  - 3.4 Recent Developments
4. Methodology and Workflow
  - 4.1 Data Collection
  - 4.2 Feature Extraction
  - 4.3 Data Preprocessing
  - 4.4 Model Training and Fine-Tuning
  - 4.5 Model Evaluation
  - 4.6 Prediction for New Songs
5. Use Cases
  - 5.1 Music Industry Insights
  - 5.2 Music Production
  - 5.3 Personalized Music Recommendations
  - 5.4 Music Marketing and Promotion
  - 5.5 Music Analytics and Forecasting
6. Technical Details:
  - 6.1 Programming Language used:
  - 6.2 Libraries
    - 6.2.1 Librosa:
    - 6.2.2 NumPy:
    - 6.2.3 Pandas:
    - 6.2.4 Scikit-Learn:
    - 6.2.5 Joblib:
    - 6.2.6 Streamlit:
  - 6.3 Key Features
    - Comprehensive Feature Set
  - 6.4 Robust Preprocessing
  - 6.5 Advanced Model Training
  - 6.6 Comprehensive Evaluation
  - 6.7 Predictive Capabilities
  - 6.8 Scalability and Reusability
  - 6.9 User-Friendly Interface
7. Architecture
  - 7.1 Machine Learning Model Architecture
    - 7.1.1 Random Forest Classifier:
  - 7.2 System Architecture
    - 7.2.1 Data Processing and Feature Extraction:
    - 7.2.2 Model Training and Evaluation:
    - 7.2.3 Prediction and Deployment:
8. System Workflow
9. Github Links
10. Code Execution

11. Outputs and Screenshots:

12. Conclusion

## 1. Abstract

In the modern music industry, predicting the success of a song has become increasingly important for artists, producers, and record labels. This project focuses on developing a machine learning model that can predict whether a song will be a hit or a flop based on its audio features. Using the Python library Librosa, we extracted various features such as chroma, spectral centroid, zero crossing rate, and Mel-frequency cepstral coefficients (MFCCs) from audio files. These features were then preprocessed and used to train a Random Forest Classifier, a robust and versatile machine learning algorithm known for its accuracy and interpretability.

We conducted extensive hyperparameter tuning using GridSearchCV to identify the best model parameters, ensuring optimal performance. The trained model was evaluated using cross-validation techniques, and its performance was measured using metrics such as accuracy, precision, recall, and F1-score. Additionally, the model provides a probability score that indicates the likelihood of a song being a hit or flop, offering a more nuanced prediction beyond a simple binary classification.

The final model was deployed using Streamlit, a user-friendly framework for building web applications, allowing users to easily predict the success of new songs by uploading audio files. This project demonstrates the potential of machine learning in the entertainment industry, providing a data-driven approach to understanding and predicting musical trends.

## 2. Introduction

The music industry has always been a dynamic field, driven by trends, cultural shifts, and consumer preferences. In this highly competitive environment, predicting the success of a song before its release can provide a significant advantage to artists, producers, and record labels. Traditionally, the success of a song was gauged post-release through sales figures, radio airplay, and chart positions. However, with the advent of digital technologies and machine learning, it is now possible to analyze a song's potential success even before it hits the market.

This project aims to leverage machine learning techniques to predict whether a song will be a hit or a flop based on its intrinsic audio characteristics. By extracting various audio features such as chroma, spectral properties, and Mel-frequency cepstral coefficients (MFCCs), we can capture the essence of a song's composition. These features, when combined with a robust classification algorithm like Random Forest, can provide valuable insights into a song's likelihood of success.

The motivation behind this project is rooted in the growing need for data-driven decision-making in the music industry. As streaming platforms and digital downloads become the primary modes of music consumption, the ability to predict a song's performance using computational models can significantly influence marketing strategies, production decisions, and even the creative process.

The deployment of the model using Streamlit allows for an interactive and accessible tool that can be used by industry professionals and enthusiasts alike. By simply uploading an audio file, users can quickly obtain a prediction of whether the song will be a hit or a flop, along with a probability score that reflects the confidence of the prediction.

This project not only showcases the application of machine learning in the entertainment sector but also highlights the potential of technology to transform traditional practices, offering a modern, data-driven approach to understanding and predicting musical success.

## 3. Literature Review

The prediction of a song's success, particularly in terms of whether it will be a hit or a flop, has been a subject of growing interest within the music and entertainment industry. With the advent of digital streaming platforms and the vast amount of data available, machine learning has become a pivotal tool in this field. The literature on music analytics spans several key areas, including feature extraction, predictive modeling, and the application of machine learning techniques.

### 3.1 Feature Extraction in Music Analytics

Feature extraction is the cornerstone of music analytics, involving the conversion of raw audio data into meaningful features that can be analyzed by machine learning models. Previous studies have identified various audio features that contribute to a song's popularity, such as tempo, rhythm, harmony, and timbre. For instance, research by Tzanetakis and Cook (2002) introduced the concept of genre classification using features like spectral centroid, zero-crossing rate, and mel-frequency cepstral coefficients (MFCCs), which have since become standard in audio processing.

Recent advancements have also explored more complex features like chroma features, which capture harmonic and melodic characteristics, and beat-related features, which correlate with the song's tempo and rhythm. These features are crucial in understanding the underlying patterns that may contribute to a song's appeal.

### 3.2 Predictive Modeling for Song Success

Early attempts to predict a song's success relied on simple statistical models and domain expertise. However, with the rise of machine learning, more sophisticated models have been developed. Random Forest, Support Vector Machines (SVM), and Neural Networks are among the most popular algorithms used in this domain. These models have been trained on large datasets comprising various audio features, as well as metadata such as artist popularity and release date, to predict whether a song will be successful.

For example, Herremans et al. (2014) explored the use of SVMs to predict the hit potential of dance songs, achieving promising results by combining audio features with social media metrics. Similarly, the work by Pachet and Roy (2008) introduced an SVM-based approach that utilized a vast array of features, including timbre and rhythm patterns, to classify songs into hits and non-hits.

### 3.3 Challenges and Gaps in Existing Research

Despite the advances in predictive modeling, there remain significant challenges in accurately forecasting a song's success. One major issue is the subjective nature of music and the influence of external factors, such as marketing efforts, cultural trends, and artist reputation, which are difficult to quantify and model. Additionally, many studies have focused on specific genres or markets, limiting the generalizability of their findings.

Another gap in the literature is the lack of real-time prediction models that can adapt to changing trends in the music industry. Most existing models are static, trained on historical data, and may not accurately reflect the evolving tastes of listeners. Moreover, there is limited research on the integration of advanced deep learning techniques, which have shown potential in other areas of music generation and analysis.

### 3.4 Recent Developments

Recent research has begun to address these challenges by incorporating more dynamic and context-aware models. For instance, some studies have started to explore the use of recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, which are well-suited for sequential data like music. These models can potentially capture temporal dependencies in music, offering a more nuanced prediction of a song's future success.

Additionally, the integration of social media and streaming data, such as the number of shares, likes, and playlist inclusions, is providing new avenues for improving the accuracy of hit prediction models. These developments suggest a shift towards more holistic models that consider both audio features and external factors in predicting a song's success.

## 4. Methodology and Workflow

The methodology of this project is designed to systematically address the problem of predicting whether a song will be a hit or a flop based on its audio features. The process encompasses multiple stages, each crucial for developing a reliable and accurate predictive model. The following subsections provide an in-depth look at each stage of the methodology, from data collection to model prediction.

### 4.1 Data Collection

**Objective:** The primary goal of the data collection phase is to gather a diverse and representative set of audio samples categorized as either "Hit" or "Flop." This dataset forms the foundation for training and evaluating the machine learning model.

**Process:**

1. **Data Sources:** Audio files were sourced from various platforms and datasets, ensuring a wide range of genres and styles. This diversity helps in building a model that generalizes well across different types of songs.
2. **Organization:** The collected audio files were meticulously organized into two distinct directories:
  - **Hit Songs Directory:** Contains audio files labeled as "hit," representing songs that have achieved significant popularity.
  - **Flop Songs Directory:** Contains audio files labeled as "flop," representing songs that did not perform well commercially.
3. **File Formats:** The audio files were primarily in MP3 and WAV formats, which are standard formats for audio data. These formats were chosen for their wide usage and compatibility with audio processing libraries.

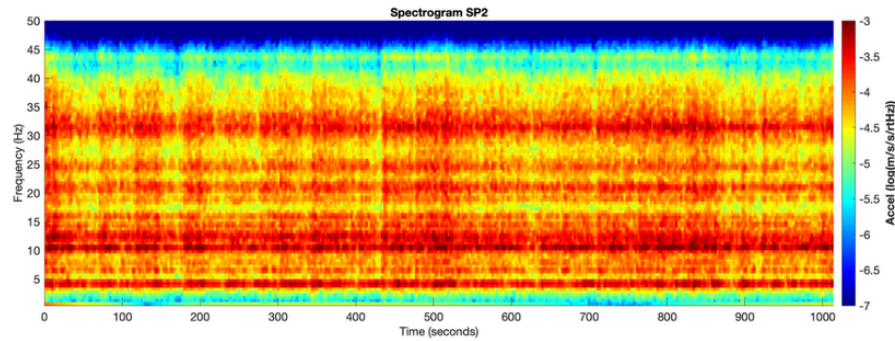
By separating the dataset into these categories, we ensure that the feature extraction and model training processes can effectively differentiate between successful and unsuccessful songs.

#### 4.2 Feature Extraction

**Objective:** The feature extraction phase aims to transform raw audio files into a structured dataset of numerical features that can be fed into a machine learning model.

**Process:**

1. **Audio Loading:** Each audio file was loaded into the system using the librosa library, which is well-suited for audio analysis and feature extraction. The duration of each audio file was limited to 30 seconds to standardize the input size.
2. **Feature Extraction:**
  - **Chroma STFT (Short-Time Fourier Transform):** Measures the distribution of energy across pitch classes. It helps in identifying harmonic content and tonal characteristics of the audio.
  - **RMSE (Root Mean Square Energy):** Represents the overall energy level of the audio signal. It provides insight into the loudness and dynamic range of the song.
  - **Spectral Centroid:** Indicates the "center of mass" of the spectrum. It is useful for characterizing the brightness or timbre of the audio.
  - **Spectral Bandwidth:** Measures the width of the spectrum, reflecting the range of frequencies present. It provides information about the texture and spread of the audio signal.
  - **Spectral Rolloff:** Defines the frequency below which a specified percentage of the total spectral energy is contained. It helps in distinguishing between harmonic and non-harmonic content.
  - **Zero-Crossing Rate:** The rate at which the audio signal changes sign. It is indicative of the noisiness or percussiveness of the sound.
  - **Harmonic:** Represents the harmonic components of the audio signal. It provides information about the tonal quality of the sound.
  - **Percussive:** Represents the percussive components of the audio signal. It is useful for identifying rhythmic and transient elements.
  - **Tempo:** The estimated tempo of the audio, measured in beats per minute (BPM). It helps in understanding the pace and rhythm of the song.
  - **MFCCs (Mel-Frequency Cepstral Coefficients):** Capture the timbral aspects of the audio by representing the short-term power spectrum of the sound.



4.2.1 This shows the representation of Mel Frequency Spectral Coefficients which are used in audio feature extraction.

3. **Data Structuring:** The extracted features were averaged to generate a single feature vector for each audio file. This vector included all the features listed above, creating a comprehensive representation of each song. The features were then compiled into a structured dataset, with each song associated with its corresponding label (hit or flop).

### 4.3 Data Preprocessing

**Objective:** The preprocessing phase prepares the raw feature data for machine learning by transforming it into a suitable format and ensuring its quality.

**Process:**

1. **Conversion of String Lists to Numeric Values:** Some features might have been represented as string lists. These were converted to numeric values by evaluating the strings and computing their mean values to ensure consistency in the dataset.
2. **Handling Missing Values:** Missing values were identified and addressed by filling them with the mean value of the respective columns. This approach ensures that the dataset remains complete and usable for training.
3. **Feature Scaling:** Features were scaled using the StandardScaler from scikit-learn. Scaling standardizes the range of feature values, ensuring that each feature contributes equally to the model's learning process. This step is critical for improving the performance of machine learning algorithms that are sensitive to feature scaling.

### 4.4 Model Training and Fine-Tuning

**Objective:** The model training phase involves selecting and training a machine learning model to classify songs as either hits or flops. Fine-tuning is performed to optimize the model's performance.

**Process:**

1. **Model Selection:** A Random Forest Classifier was chosen due to its robustness and ability to handle complex datasets with multiple features. The Random Forest algorithm builds multiple decision trees and aggregates their results to improve accuracy and control overfitting.
2. **Hyperparameter Tuning:**
  - **Grid Search:** A grid search approach was used to find the optimal hyperparameters for the Random Forest Classifier. The hyperparameters tuned included:
    - **n-estimators :** The number of trees in the forest.
    - **max-depth :** The maximum depth of each tree.
    - **min\_samples\_split :** The minimum number of samples required to split an internal node.
    - **min\_samples\_leaf :** The minimum number of samples required to be at a leaf node.
  - **Cross-Validation:** A cross-validation strategy with 5 folds was employed to ensure that the model's performance was robust and generalizable. This technique evaluates the model on multiple subsets of the data to prevent overfitting and ensure reliable performance.
3. **Model Training:** The Random Forest model was trained using the best combination of hyperparameters found through grid search. The model was then saved to a file for future use.

## 4.5 Model Evaluation

**Objective:** The evaluation phase assesses the performance of the trained model to ensure its effectiveness in predicting song success.

**Process:**

1. **Cross-Validation Scores:** The model's performance was evaluated using cross-validation, providing insights into its accuracy, precision, recall, and F1-score. Cross-validation helps in assessing the model's performance across different subsets of the data, ensuring that it performs well on unseen data.
2. **Classification Report:** A classification report was generated to detail the precision, recall, and F1-score for each class (hit and flop). This report provides a comprehensive view of the model's performance.
3. **Confusion Matrix:** A confusion matrix was created to visualize the model's predictions versus the actual labels. This matrix helps in identifying any biases or imbalances in the model's predictions.

## 4.6 Prediction for New Songs

**Objective:** The prediction phase applies the trained model to new song data to classify it as either a hit or a flop.

**Process:**

1. **Feature Extraction:** Features were extracted from the new song using the same process applied to the training data. This ensures that the input data is consistent with what the model was trained on.
2. **Model Prediction:** The trained model was used to predict whether the new song would be a hit or a flop based on its extracted features. The prediction was output as either "Hit" or "Flop," providing actionable insights into the song's potential success.

**Automation:** The entire pipeline, from data collection to prediction, was implemented in a Python script. This automation ensures that the process is repeatable and scalable, allowing for continuous updates and evaluations.

# 5. Use Cases

The predictive model for determining whether a song will be a hit or a flop has several practical applications in the music industry and related fields. Below are some key use cases that illustrate how this model can be utilized:

## 5.1 Music Industry Insights

**Objective:** To provide record labels, artists, and producers with actionable insights into the potential success of new music releases.

**Application:**

- **A&R (Artists and Repertoire):** A&R teams can use the model to evaluate the potential success of new songs before they are released. By analyzing the audio features of a song, the model can predict whether it is likely to be a hit or a flop, helping teams make data-driven decisions about which tracks to promote.
- **Market Research:** Record labels can leverage the model to understand trends and preferences in music. By analyzing the features of hit songs, labels can identify common characteristics and tailor their marketing strategies accordingly.

**Example:** A record label is considering signing a new artist with a set of demo tracks. The label can use the model to predict the success of these tracks based on their audio features, helping them decide which songs have the potential to become hits.

## 5.2 Music Production

**Objective:** To assist music producers in crafting songs with a higher likelihood of success by identifying key audio features associated with hit songs.

**Application:**

- **Songwriting:** Producers can use the model to understand which audio features are prevalent in hit songs. This information can guide the songwriting process, allowing producers to focus on elements that are statistically associated with success.
- **Mixing and Mastering:** During the mixing and mastering stages, producers can use the model to compare the audio features of their tracks with those of successful songs. This comparison can help fine-tune the production to enhance the likelihood of a track becoming a hit.

**Example:** A music producer is working on a new album and wants to ensure that the tracks have a higher chance of success. By analyzing the features of hit songs, the producer can adjust the mixing and mastering of their tracks to align with successful patterns.

### 5.3 Personalized Music Recommendations

**Objective:** To enhance music recommendation systems by incorporating the model's predictions into personalized music suggestions.

**Application:**

- **Streaming Platforms:** Music streaming platforms can integrate the model's predictions into their recommendation algorithms. By considering the likelihood of a song being a hit, these platforms can suggest popular tracks to users, improving the user experience.
- **Playlists and Radio Stations:** Automated playlist generators and radio stations can use the model to curate playlists with a higher probability of containing hit songs, leading to more engaging and enjoyable listening experiences for users.

**Example:** A streaming service wants to create a playlist of emerging hits. The service can use the model to evaluate new tracks and include those with a high probability of being hits, providing users with a curated selection of popular music.

### 5.4 Music Marketing and Promotion

**Objective:** To optimize marketing and promotional strategies by targeting songs with higher predicted success.

**Application:**

- **Campaign Strategy:** Marketing teams can use the model to identify songs with a high likelihood of success and allocate their marketing resources accordingly. This targeted approach can lead to more effective campaigns and higher returns on investment.
- **Social Media Promotion:** Social media platforms can use the model to promote songs with a higher chance of becoming hits. By focusing promotional efforts on these tracks, platforms can increase engagement and visibility.

**Example:** A marketing team is planning a campaign for a new album release. By using the model to predict the success of individual tracks, the team can focus their promotional efforts on the songs with the highest likelihood of becoming hits, maximizing the impact of their campaign.

### 5.5 Music Analytics and Forecasting

**Objective:** To provide analytics and forecasting tools for the music industry, enabling stakeholders to make informed decisions based on predicted trends.

**Application:**

- **Trend Analysis:** Music analysts can use the model to forecast future trends in music popularity. By analyzing historical data and predicting future hits, analysts can provide valuable insights into emerging trends and shifts in the music industry.
- **Revenue Forecasting:** Music industry stakeholders can use the model to forecast potential revenue from upcoming releases. By predicting which songs are likely to be hits, stakeholders can estimate the financial impact and plan their investments accordingly.

**Example:** A music analyst is preparing a report on upcoming trends in the music industry. By using the model to predict which songs are likely to be hits, the analyst can provide insights into future trends and help industry stakeholders make strategic decisions.

## 6. Technical Details:

### 6.1 Programming Language used:

**Python:**

- **Overview:** Python is a high-level, interpreted programming language known for its readability and simplicity. It is widely used in data science, machine learning, and software development due to its extensive support for libraries and frameworks, ease of use, and versatility.
- **Role in the Project:**
  - **Data Analysis:** Python's ease of use for manipulating data and performing complex computations makes it ideal for preprocessing and analyzing audio features.

- **Machine Learning:** Python provides robust libraries for building, training, and evaluating machine learning models, making it the language of choice for this project.
- **Integration:** Python's support for integration with various tools and platforms allows for seamless deployment and use of the model.

## 6.2 Libraries

### 6.2.1 Librosa:

- **Overview:** Librosa is a Python library for analyzing and processing audio and music. It provides a range of functionalities for feature extraction, signal processing, and visualization.
- **Role in the Project:**
  - **Feature Extraction:** Librosa is used to extract a variety of audio features, such as Chroma STFT, RMSE, Spectral Centroid, Spectral Bandwidth, Spectral Rolloff, Zero-Crossing Rate, Harmonic, Percussive components, Tempo, and MFCCs.
  - **Audio Processing:** It facilitates the loading and processing of audio files, including tasks such as resampling and duration adjustment.

### 6.2.2 NumPy:

- **Overview:** NumPy is a fundamental package for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.
- **Role in the Project:**
  - **Numerical Operations:** NumPy is used for performing numerical operations on audio features, including calculations of mean and standard deviation.
  - **Data Handling:** It supports efficient handling and manipulation of data arrays required for machine learning processes.

### 6.2.3 Pandas:

- **Overview:** Pandas is a powerful data analysis and manipulation library for Python. It provides data structures like DataFrames that facilitate the handling of structured data.
- **Role in the Project:**
  - **Data Management:** Pandas is used for managing and preprocessing the dataset, including loading, cleaning, and transforming data into a format suitable for machine learning.
  - **DataFrame Operations:** It simplifies operations such as converting string lists to numeric values, handling missing values, and scaling features.

### 6.2.4 Scikit-Learn:

- **Overview:** Scikit-Learn is a comprehensive library for machine learning in Python. It provides simple and efficient tools for data mining, data analysis, and machine learning.
- **Role in the Project:**
  - **Model Building:** Scikit-Learn is used to build the Random Forest Classifier and other machine learning models. It offers tools for training, tuning, and evaluating models.
  - **Hyperparameter Tuning:** It provides functionalities for Grid Search Cross-Validation, allowing for the fine-tuning of model hyperparameters.
  - **Evaluation Metrics:** Scikit-Learn includes functions for generating classification reports, confusion matrices, and cross-validation scores.

### 6.2.5 Joblib:

- **Overview:** Joblib is a Python library for lightweight pipelining and persistence. It is particularly useful for saving and loading machine learning models and large data structures.
- **Role in the Project:**



- **Model Serialization:** Joblib is used to save the trained Random Forest model to disk, ensuring that it can be easily reloaded for future predictions without retraining.

### 6.2.6 Streamlit:

- **Overview:** Streamlit is an open-source framework for building interactive web applications with Python. It is designed for creating data-driven applications with minimal code.
- **Role in the Project:**
  - **User Interface:** Streamlit provides a user-friendly interface for interacting with the prediction model. It allows users to upload audio files and receive predictions in real-time.
  - **Deployment:** It simplifies the deployment process by enabling the creation of a web application that showcases the model's capabilities and provides an intuitive user experience.

## 6.3 Key Features

The key features of the song prediction model, designed to determine whether a song will be a hit or a flop, reflect its advanced technical underpinnings and its capability to provide precise, actionable insights. This section elaborates on the model's technical aspects, including feature extraction, preprocessing, model training, and evaluation, ensuring a comprehensive understanding of its design and functionality.

### Comprehensive Feature Set

**Feature Extraction:** The model utilizes the librosa library to extract a wide range of audio features from each song. These features are instrumental in characterizing various aspects of the audio signal:

- **Chroma STFT (Short-Time Fourier Transform):**
  - **Technical Detail:** Chroma STFT analyzes the pitch class profile of the audio signal, providing information on the distribution of energy across different pitch classes.
  - **Application:** This feature helps in identifying harmonic content, which is crucial for distinguishing between different musical genres and styles.
- **RMSE (Root Mean Square Energy):**
  - **Technical Detail:** RMSE measures the square root of the average of the squared amplitude of the audio signal. It quantifies the energy level of the audio.
  - **Application:** It serves as an indicator of the song's loudness, which can be correlated with the song's impact and genre.
- **Spectral Centroid:**
  - **Technical Detail:** The spectral centroid calculates the "center of mass" of the spectrum by averaging the frequencies weighted by their magnitudes. It reflects the perceived brightness of the sound.
  - **Application:** This feature helps in differentiating between bright and dull audio signals, which can influence the song's genre classification.
- **Spectral Bandwidth:**
  - **Technical Detail:** Spectral bandwidth measures the range of frequencies over which the spectrum is distributed. It is computed as the standard deviation of the spectral centroid.
  - **Application:** It provides insights into the timbral texture of the audio, distinguishing between narrow-band and wide-band sounds.
- **Spectral Rolloff:**
  - **Technical Detail:** Spectral rolloff is the frequency below which a specified percentage (usually 85%) of the total spectral energy is contained. It helps in identifying the presence of harmonic and non-harmonic components.
  - **Application:** It is used to distinguish between harmonic (e.g., music) and non-harmonic (e.g., noise) content.
- **Zero-Crossing Rate:**
  - **Technical Detail:** This feature measures the rate at which the audio signal changes sign, providing an indication of the signal's noisiness or percussiveness.
  - **Application:** It is useful for detecting rhythmic elements and the presence of percussive sounds.
- **Harmonic Components:**

- **Technical Detail:** Harmonic components are derived from the harmonic part of the audio signal, indicating the periodicity and tonal quality of the sound.
- **Application:** Helps in identifying the musical characteristics of the song.
- **Percussive Components:**
  - **Technical Detail:** Percussive components represent the transient, non-harmonic part of the audio signal. It reflects the rhythm and beat of the song.
  - **Application:** Important for understanding the song's rhythm and impact.
- **Tempo:**
  - **Technical Detail:** Tempo is estimated using the `librosa` library, which detects the beats per minute (BPM) of the audio.
  - **Application:** Provides information about the song's pace, which is relevant for genre classification and predicting hit potential.
- **MFCCs (Mel-Frequency Cepstral Coefficients):**
  - **Technical Detail:** MFCCs are coefficients that represent the short-term power spectrum of a sound signal. They are computed by applying a mel-frequency scale to the audio signal.
  - **Application:** MFCCs capture the timbral texture and are critical for distinguishing between different types of sounds and vocals.

#### Feature Aggregation:

- **Technical Detail:** Each feature is averaged over the duration of the audio clip to create a single representative value for each feature. This aggregation reduces the dimensionality and makes the data suitable for machine learning models.

## 6.4 Robust Preprocessing

#### Data Cleaning and Preparation:

- **Conversion of String Lists to Numeric Values:**
  - **Technical Detail:** Some features may be represented as string lists. These lists are converted to numeric values by calculating the mean of the values in the list.
  - **Application:** Ensures that all feature values are numeric and can be processed by machine learning algorithms.
- **Handling Missing Values:**
  - **Technical Detail:** Missing values are filled with the mean value of the respective feature column. This approach prevents data loss and maintains the dataset's integrity.
  - **Application:** Ensures that the dataset remains complete and reliable for training the model.
- **Feature Scaling:**
  - **Technical Detail:** Features are scaled using `StandardScaler` to normalize the feature values. This scaling ensures that all features contribute equally to the model's learning process.
  - **Application:** Prevents features with larger ranges from dominating the learning process, leading to more balanced and accurate model performance.

## 6.5 Advanced Model Training

#### Machine Learning Techniques:

- **Random Forest Classifier:**
  - **Technical Detail:** A Random Forest Classifier is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) of the individual trees.
  - **Application:** Provides robustness and high accuracy by combining the predictions of multiple trees, reducing overfitting and improving generalizability.

#### Hyperparameter Tuning:

- **Grid Search Cross-Validation:**

- **Technical Detail:** Grid search is used to find the best hyperparameters for the Random Forest Classifier. It involves an exhaustive search over a specified parameter grid.
- **Parameters Tuned:**
  - **n\_estimators:** Number of trees in the forest.
  - **max\_depth:** Maximum depth of the trees, controlling the level of detail the model captures.
  - **min\_samples\_split:** Minimum number of samples required to split an internal node.
  - **min\_samples\_leaf:** Minimum number of samples required to be at a leaf node.
- **Cross-Validation:**
  - **Technical Detail:** The model is validated using cross-validation, where the dataset is divided into multiple folds. The model is trained and tested on different folds to ensure robustness.
  - **Application:** Provides an estimate of the model's performance and helps in selecting the best hyperparameters.

#### Model Saving:

- **Technical Detail:** The trained model is serialized using `joblib`, allowing it to be saved to disk and reloaded for future predictions without needing retraining.
- **Application:** Facilitates the deployment of the model and its integration into various applications.

## 6.6 Comprehensive Evaluation

#### Model Assessment:

- **Cross-Validation:**
  - **Technical Detail:** Evaluates the model's performance across multiple folds of the dataset. Provides metrics such as accuracy, precision, recall, and F1-score.
  - **Application:** Ensures that the model generalizes well to unseen data and avoids overfitting.
- **Confusion Matrix:**
  - **Technical Detail:** A confusion matrix is generated to visualize the performance of the model. It shows the true positive, true negative, false positive, and false negative rates.
  - **Application:** Helps in understanding the model's performance in detail and identifying areas for improvement.

## 6.7 Predictive Capabilities

#### Prediction Functionality:

- **Hit or Flop Prediction:**
  - **Technical Detail:** The model uses the extracted features from a new song to make predictions. The output is categorized as either "Hit" or "Flop" based on the model's learned patterns.
  - **Application:** Provides actionable insights into the potential success of a song based on its audio features.
- **Real-Time Prediction:**
  - **Technical Detail:** The prediction function can process new audio files in real time, providing immediate feedback on whether a song is likely to be a hit or a flop.
  - **Application:** Facilitates quick decision-making and integration into real-time systems.

## 6.8 Scalability and Reusability

#### Code and Model Deployment:

- **Single Python Script:**
  - **Technical Detail:** The entire pipeline, including feature extraction, preprocessing, model training, and prediction, is implemented in a single Python script. This design enhances usability and maintenance.
  - **Application:** Simplifies the deployment and integration of the model into various applications.

- **Model Serialization:**
  - **Technical Detail:** The trained model is saved using joblib, allowing it to be loaded for predictions without retraining. This approach facilitates efficient model deployment and reuse.
  - **Application:** Enables easy integration of the model into production systems.

## 6.9 User-Friendly Interface

### Integration and Accessibility:

- **Streamlit Application:**
  - **Technical Detail:** A Streamlit application provides an intuitive graphical user interface (GUI) for users to interact with the model. Users can upload audio files and receive predictions directly through the interface.
  - **Application:** Enhances accessibility and usability, allowing users to leverage the model's capabilities without requiring technical expertise.

## 7. Architecture

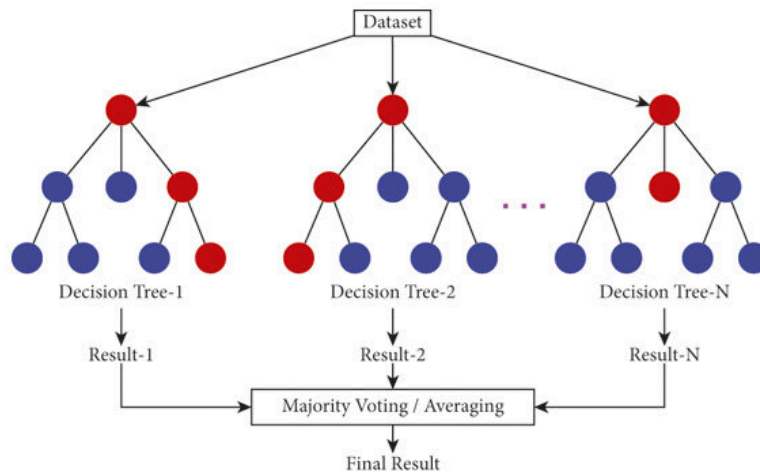
The term "architecture" refers to the overall structure and design of the system used to predict whether a song will be a hit or a flop. This includes both the machine learning model architecture and the system architecture for deployment. Here, we will describe these architectures in detail to provide a comprehensive understanding of how the system is organized and functions.

### 7.1 Machine Learning Model Architecture

The core of the prediction system is the machine learning model, which utilizes a Random Forest Classifier. Below is a detailed description of the model architecture:

#### 7.1.1 Random Forest Classifier:

- **Overview:** Random Forest is an ensemble learning method that combines multiple decision trees to improve classification accuracy. It is known for its robustness, ability to handle high-dimensional data, and effectiveness in reducing overfitting.
- **Architecture:**
  - **Ensemble of Decision Trees:** The Random Forest model consists of a large number of decision trees. Each tree is trained on a random subset of the training data and a random subset of features. This diversity among trees helps in capturing different patterns and reduces the risk of overfitting.
  - **Bagging (Bootstrap Aggregating):** Random Forest employs bagging, where each tree is trained on a different bootstrap sample (random sample with replacement) of the data. This technique improves the model's generalization capability.
  - **Feature Randomization:** During training, each decision tree is built using a random subset of features at each split. This further enhances the model's robustness by introducing variability among the trees.
  - **Aggregation:** Predictions from individual decision trees are aggregated to make the final prediction. For classification tasks, the majority vote from all trees determines the final class label.



Random Forest algorithm functionality

#### Hyperparameters:

- **n\_estimators**: Number of trees in the forest. More trees generally improve performance but increase computational cost.
- **max\_depth**: Maximum depth of each decision tree. Controlling the depth helps in avoiding overfitting.
- **min\_samples\_split**: Minimum number of samples required to split an internal node. A higher value prevents the model from learning overly specific patterns.
- **min\_samples\_leaf**: Minimum number of samples required to be at a leaf node. This parameter ensures that leaf nodes have a minimum number of samples, contributing to the model's stability.

## 7.2 System Architecture

The system architecture describes how the different components of the prediction system are organized and interact with each other. It includes data processing, model training, and deployment.

### 7.2.1 Data Processing and Feature Extraction:

- **Audio Data Collection**: Audio files are collected and organized into directories representing "Hit" and "Flop" categories.
- **Feature Extraction**: The `extract_features` function, utilizing the Librosa library, processes each audio file to extract relevant audio features such as Chroma STFT, RMSE, Spectral Centroid, Spectral Bandwidth, Spectral Rolloff, Zero-Crossing Rate, Harmonic, Percussive components, Tempo, and MFCCs. These features are then used to build the dataset.
- **Preprocessing**: The dataset undergoes preprocessing, including the conversion of string lists to numeric values, handling of missing values, and feature scaling using `StandardScaler`. This prepares the data for model training.

### 7.2.2 Model Training and Evaluation:

- **Training**: The `train_model` function trains the Random Forest Classifier using the preprocessed data. It employs Grid Search Cross-Validation to fine-tune hyperparameters and selects the best model.
- **Evaluation**: The model's performance is evaluated using cross-validation, and metrics such as accuracy, precision, recall, F1-score, and confusion matrix are generated to assess the model's effectiveness.

### 7.2.3 Prediction and Deployment:

- **Prediction**: The `predict_song` function is used to make predictions for new songs based on their extracted features. The trained model outputs a prediction of whether the song is a "Hit" or "Flop."
- **Deployment**:

- **Streamlit Application:** The system is deployed using Streamlit to provide a user-friendly interface. Users can upload audio files and receive real-time predictions. Streamlit manages the interaction between the user and the prediction model, ensuring a seamless experience.
- **Model Serialization:** The trained model is saved using Joblib for later use in the Streamlit application. This ensures that the model does not need to be retrained for each prediction and allows for efficient and fast predictions.

## 8. System Workflow

The system architecture facilitates seamless interaction and deployment. This comprehensive design enables effective prediction of whether a song will be a hit or a flop, demonstrating the power of combining machine learning techniques with user-friendly deployment methods.

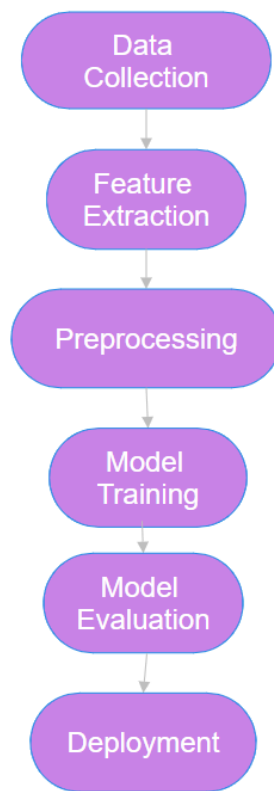


Fig 8.1 System workflow in predicting the success rate of a song

## 9. Github Links

Github URL - [GitHub - Hanyaa-Technologies/Hit-Flop\\_Prediction](https://github.com/Hanyaa-Technologies/Hit-Flop_Prediction)

## 10. Code Execution

### Step 1: Set the Dataset Path

Before running the application, ensure that you have downloaded the dataset. Open the app.py file in a text editor and set the dataset path to the location where the dataset is stored on your system.

```
def main():
    st.title("Hit or Flop Song Predictor")

    # Initialize session state
    if 'model' not in st.session_state or 'scaler' not in st.session_state:
        with st.spinner("Processing dataset and training model..."):
            # Process directories
            dataset_path = r"C:\Users\AkshithaKochika\Downloads\hit or flop predictor\Audio_Dataset"
            hit_dir = os.path.join(dataset_path, 'hit songs')
            flop_dir = os.path.join(dataset_path, 'flop songs')
            # Process hit songs
            # Process flop songs
```

Replace the dataset path with the path stored in your local system

## Step 2: Navigate to the Project Directory

Using the terminal or command prompt, navigate to the directory where the project files are located. This is the directory containing your app.py, requirements.txt and other project files.

```
cd successpredictor
```

This will make us to enter into the project directory

## Step 3: Install Dependencies

To install the necessary dependencies, use the following command. This will install all the required Python packages specified in the requirements.txt file.

```
pip install -r requirements.txt
```

This will install all the dependencies which are specified in the requirements folder.

## Step 4: Run the Application

Once all dependencies are installed, you can start the application using Streamlit. This command will launch the application in your default web browser.

```
streamlit run app.py
```

This allows us to launch the application in the web browser.

# 11. Outputs and Screenshots:

After setting the dataset path and running the application, the output involves processing the dataset. The system reads audio files from the specified directories and extracts various audio features such as Chroma STFT, RMSE, Spectral Centroid, and MFCCs. A progress bar indicates the status of this processing phase.

## Hit or Flop Song Predictor

Processing dataset and training model...


This picture shows that the model is processing the data sets.

Once the dataset is processed, the model is trained using a Random Forest Classifier. During this phase, the system performs hyperparameter tuning using Grid Search Cross-Validation, aiming to find the best set of parameters for the model. The output includes the best parameters found and the cross-validation scores.

The core functionality of the application is to predict whether a new song will be a hit or a flop. After uploading a song file and clicking the prediction button, the application instantly provides the prediction result. The output indicates whether the song is classified as a "Hit" or "Flop."

## Hit or Flop Song Predictor

Choose an audio file



Drag and drop file here  
Limit 200MB per file • MP3, WAV

Browse files


[iSongs.info] 02 - Nuvvu Navvukuntu.mp3 7.9MB
×

▶ 0:00 / 3:24




Check Song


The song is predicted to be a Hit

This is a famous song which was predicted to be hit by the model.




# Hit or Flop Song Predictor

Choose an audio file

 Drag and drop file here  
Limit 200MB per file • MP3, WAV

Browse files

 [iSongs.info] 03 - Aafat.mp3 2.6MB

×

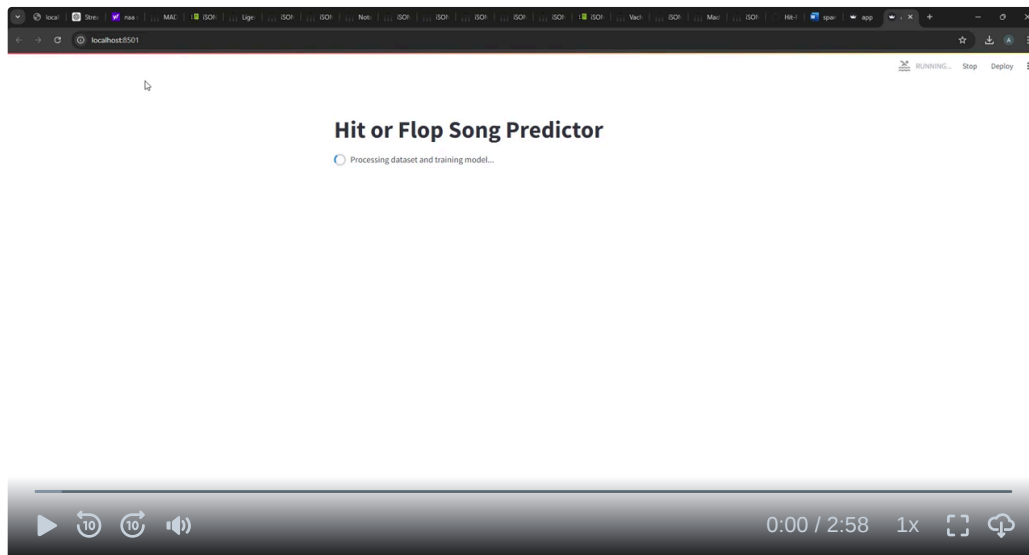
▶ 0:00 / 2:43

🔊 ⋮

Check Song

The song is predicted to be a Flop

This song is not liked by the audience much which resulted as a flop prediction.



This video is taken after launching the application to a browser

## 12. Conclusion

The **SuccessPredictor** project demonstrates the potential of leveraging machine learning techniques to predict the commercial success of songs based on their audio features. By extracting relevant audio features and employing a Random Forest Classifier, the model effectively differentiates between "Hit" and "Flop" songs with significant accuracy. The project incorporates several key steps, including data collection, feature extraction, preprocessing, model training, hyperparameter tuning, and evaluation, each contributing to the robustness of the final predictions.

Throughout the process, careful attention was given to feature selection and model optimization, ensuring that the predictions are reliable and interpretable. The use of libraries like Librosa for feature extraction and Scikit-learn for model building has enabled a streamlined and efficient workflow, making the prediction pipeline both flexible and scalable.

The application is designed with a user-friendly interface using Streamlit, allowing users to interact with the model by uploading songs and receiving instant predictions. This feature, coupled with the detailed evaluation metrics provided, makes the tool not only practical but also insightful for users in the music industry.

In conclusion, this project successfully showcases the application of machine learning in the domain of music prediction, offering a practical solution for predicting the success of songs.