A PROJECT REPORT

On

"MUSIC RECOMMENDATION SYSTEM" (Mood Sonic)

Submitted to
KIIT Deemed to be University

In Partial Fulfillment of the Requirement for the Award of
BACHELOR'S DEGREE IN
COMPUTER SCIENCE

BY

JHANVI JAIN	**21053233**
ANKITA KUMARI	**21053235**
RAJBIR SINGH	**21053357**
SOMYA SINHA	**21053365**

UNDER THE GUIDANCE OF
MR. VISHAL MEENA

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBNESWAR, ODISHA - 751024

May 2024

CERTIFICATE

This is certify that the project entitled

"MUSIC RECOMMENDATION SYSTEMS"
(Mood Sonic)

Submitted by

JHANVI JAIN	21053233
ANKITA KUMARI	21053235
RAJBIR SINGH	21053357
SOMYA SINHA	21053365

Is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science at KIIT Deemed to be university, Bhubneswar. This work is done during year 2024-2025, under our guidance.

Date:12/04/24

MR. VISHAL MEENA

Acknowledgement

I would like to express my sincere gratitude to Professor Vishal Meena of the Department of CSE at Kalinga Institute of Industrial Technology. His deep knowledge and keen interest in Machine Learning provided invaluable guidance throughout this project. Professor Meena's constant support, patience, and constructive criticism were instrumental in shaping this work. I am particularly grateful for his willingness to read and provide feedback on multiple drafts.

This project would not have been possible without the collaboration and contributions of my fellow group members. I truly appreciate their willingness to help each other throughout the process.

I would also like to acknowledge the supportive environment provided by the Kalinga Institute staff, both teaching and non-teaching. Their assistance, whenever needed, is greatly appreciated.

Finally, I am incredibly grateful for the unwavering encouragement and patience of my parents. Their constant support has been a source of strength throughout my academic journey.

JHANVI JAIN	21053233
ANKITA KAUMRI	21053235
RAJBIR SINGH	21053357
SOMYA SINHA	21053365

Abstract

The explosion of music online has fueled the rise of recommendation systems. Just like how Amazon and Flipkart leverage them in e-commerce, or Wynk and Gaana in music streaming, these systems connect users with content they might enjoy. This project proposes a framework specifically for music recommendations, aiming to suggest new songs based on user preferences.

Music holds a special place in many lives, offering a powerful emotional connection and a way to unwind. Recognizing the growing demand for recommendation systems, we saw an opportunity to create a tool that helps music lovers discover new favorites. This report outlines the project's purpose and the progress made thus far. It details the implemented techniques, tools, and current stage of development, along with project timelines and deliverables.

The core objective of this music recommendation system is to prioritize user experience by delivering personalized and relevant suggestions. Unlike some systems that solely rely on user listening history or song characteristics, our framework is designed to adapt to other factors that might influence music preferences.

Keywords: Music Recommendation System, Machine Learning, Personalization, User Profiling, Data Mining.

Contents

Chapter - 1

Introduction

The development of recommendation systems has accelerated since their introduction in the 1990s, fueled by Machine Learning and powerful computing networks. This growth coincides with the explosion of data in today's digital world. The vast user databases we can now collect would be impossible to analyze thoroughly without Machine Learning's capabilities.

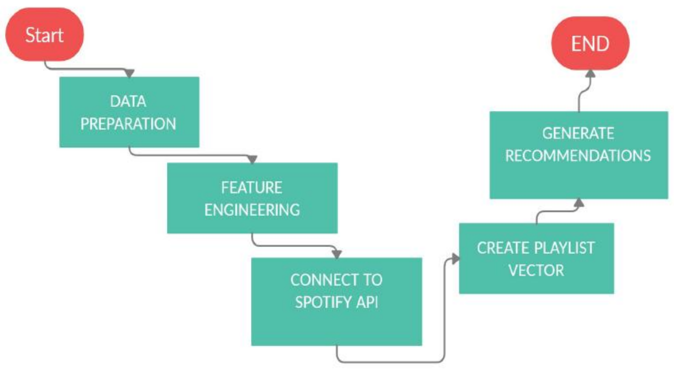
Unlike search engines where users provide specific queries, recommendation systems proactively suggest new items based on past behavior. This is crucial for e-commerce platforms aiming to boost sales or streaming services striving to retain users. The key lies in delivering relevant recommendations that simplify decision-making and save users time.

Recommendation systems have become increasingly valuable in the music industry, especially after the shift towards subscription and streaming models. With a vast library of digital music readily available, helping users navigate this abundance becomes crucial. Here, music recommendation systems play a predictive role.

Traditional recommendation systems often rely on user behavior (collaborative filtering) or music characteristics (content-based filtering). This project aims to explore the potential of incorporating real-time data - such as a user's heart rate and time of day - to provide more personalized suggestions.

We propose a mobile application that leverages Machine Learning techniques to generate recommendations. The app will integrate with a smartwatch to monitor the user's heart rate in real-time. This data, along with the time of day, will be used to suggest music typically associated with that user's heart rate and time preferences.
Many music platforms have adopted recommendation algorithms. The transition from physical music sales to cloud-based streaming has created vast music libraries. Our system aims to categorize these songs based on genre, artist region, age group, and language to better align with individual user preferences.

Understanding user behavior and delivering valuable services are key to attracting a large user base. This project utilizes various Machine Learning and data mining techniques to achieve this. We will evaluate different algorithms and choose the one that provides the most effective recommendations for our model.



Chapter 2

Basic Concepts/ Literature Review

In this section, we delve into the fundamental concepts and relevant literature that underpin the tools and techniques employed in MoodSonic, a music recommendation system focusing on emotional content.

2.1 Machine Learning Models

Machine learning models are pivotal in the functionality of MoodSonic, allowing it to discern and interpret the emotional characteristics of songs. These models, ranging from traditional algorithms like decision trees and support vector machines to more advanced techniques such as neural networks and gradient boosting, enable MoodSonic to analyze large datasets efficiently and make accurate predictions. Through supervised learning, these models are trained on labeled song data, where emotional attributes such as happiness, sadness, calmness, and energy are associated with each song. By recognizing patterns in the features extracted from songs, machine learning models can map these features to corresponding emotional states, facilitating personalized music recommendations for users.

2.2 Feature Engineering

Feature engineering plays a crucial role in extracting relevant information from raw musical data. In the context of MoodSonic, features such as acousticness, danceability, energy, and tempo are extracted from each song to capture its emotional essence. Techniques such as statistical analysis, dimensionality reduction, and normalization are applied to preprocess the data and transform it into a format suitable for machine learning algorithms. By selecting and engineering meaningful features, MoodSonic enhances the performance and interpretability of its recommendation system, ultimately improving the user experience.

2.3 Spotify API

The Spotify API serves as a valuable resource for MoodSonic, providing access to a vast repository of songs and their associated metadata. Through the API, MoodSonic can retrieve information such as track features, artist details, and user playlists, facilitating seamless integration with the Spotify platform. This integration enables users to access personalized recommendations directly within the Spotify interface, enhancing the overall user experience and increasing engagement with the MoodSonic system.

2.4 Emotional Classification Models

Emotional classification models form the core of MoodSonic's recommendation system, enabling the system to categorize songs based on their emotional content. These models leverage techniques such as natural language processing (NLP), sentiment analysis, and machine learning to analyze song lyrics, audio features, and user-generated content to infer emotional states. By understanding the emotional nuances embedded within songs, MoodSonic can generate recommendations that resonate with users' current moods, enhancing their listening experience.

2.5 User Interaction and Feedback Mechanisms

User interaction and feedback mechanisms are essential components of MoodSonic, allowing users to provide input and refine their music recommendations over time. Through features such as like/dislike buttons, user ratings, and personalized playlists, MoodSonic gathers feedback from users and adapts its recommendations accordingly. By incorporating user preferences and listening habits into its recommendation algorithm, MoodSonic ensures that users receive relevant and personalized music suggestions that align with their individual tastes and preferences.

2.6 Related Work and Existing Literature

Several studies and research papers have explored the intersection of music and emotion, providing valuable insights into the psychological, physiological, and cognitive mechanisms underlying emotional processing in music. Works by scholars such as Juslin and Sloboda (2001), Eerola and Vuoskoski (2013), and McFee and Lanckriet (2012) have investigated various aspects of music perception, emotion induction, and personalized music recommendation. By drawing on this existing literature, MoodSonic aims to build upon prior knowledge and contribute to the advancement of music recommendation technology, ultimately enhancing the emotional well-being and satisfaction of its users.

Chapter 3

Problem Statement / Requirement Specifications

MoodSonic aims to enhance music recommendation systems by leveraging emotion recognition in songs. The project utilizes the LightGBM (LGBM) model to classify songs into four distinct emotions: happiness, sadness, calmness, and energy. By incorporating emotion analysis and cosine similarity, MoodSonic seeks to generate personalized recommendations based on the emotional context of songs.

3.1 Project Planning

1. Data Collection and Preparation:

- Gather data from Spotify API, focusing on songs from extensively listened-to playlists and user-curated selections.
- Ensure balanced representation of diverse moods: happiness, sadness, calmness, and energy.
- Extract relevant features such as acousticness, danceability, energy, etc., from the retrieved songs.

2. Exploratory Data Analysis (EDA) and Feature Engineering:

- Conduct EDA to understand the distribution of features across different emotion.
- Explore correlations between features and emotions.
- Handle missing values, outliers, and perform necessary data transformations.
- Engineer new features if required to improve model performance.

3. Machine Learning Model Development:

- Select appropriate machine learning algorithms for emotion recognition, considering LGBM (Light Gradient Boosting Machine) as one of the options.
- Split the dataset into training and testing sets.
- Train the models using training data.
- Evaluate model performance using appropriate metrics such as accuracy, precision, recall, and F1-score.
- Choose the best-performing model for emotion recognition.

4. Hyper-parameter Optimization:

- Perform hyperparameter tuning to fine-tune the selected machine learning model.
- Utilize techniques like grid search or random search to optimize hyperparameters.
- Validate the optimized model to ensure improved performance.

5. Integration of Emotion Recognition with Cosine Similarity:

- Implement cosine similarity to generate emotion-focused recommendations based on the recognized emotions of songs.
- Combine the obtained emotion output with cosine similarity to enhance recommendation accuracy.
- Validate the effectiveness of combined recommendation approach.

6. Development of Streamlit App for Deployment:

- Develop a user-friendly Streamlit application for the MoodSonic project.
- Integrate the trained model and recommendation system into the app.
- Ensure smooth functionality and user experience.
- Test the app thoroughly to identify and resolve any issues.

7. Documentation and Presentation:

- Document the entire project including data collection methods, EDA findings, model development, and app deployment steps.
- Create a presentation summarizing the project goals, methodology, findings, and potential future improvements.
- Ensure clear and comprehensive documentation for easy understanding and replication by other users or developers.

8. Testing and Validation:

- Conduct thorough testing of all components of the project to ensure functionality and accuracy.
- Validate the emotion recognition model and recommendation system with real user interactions.
- Gather feedback from users for further improvements.

9. Deployment and Maintenance:

- Deploy the MoodSonic app on a suitable platform for public access.
- Monitor app performance and user feedback regularly.
- Address any issues or bugs promptly.
- Plan for future updates and maintenance to keep the app relevant and functional.

- By following these steps, you can effectively plan and execute the development of the MoodSonic project, ensuring a robust and user-friendly application for emotion-based music recommendations.

3.2 Project Analysis

3.2.1. Functional Requirements:

- Functional requirements for the project are divided into user needs, security requirements, and device requirements.

- Under user needs, one requirement states that users must be registered on the platform and have listened to at least one song to access music suggestions.

- This condition ensures that users have engaged with the platform before exploring music recommendations.

- The requirement aims to personalize suggestions based on user activity and preferences.

- By having users register and listen to music, the platform can gather data to improve recommendation accuracy.
- Implementing this requirement enhances user experience by tailoring suggestions to individual tastes.
-
- It also serves as a security measure to ensure only legitimate users access the music recommendation feature.

- Overall, this user requirement contributes to the functionality and security of the platform.

3.2.2. Non-Functional Requirements:

- i. Performance: The system is designed to deliver prompt, accurate, and reliable results.
-
- ii. Capacity and Scalability: The system will have the capability to store all registered identifications efficiently within the database.
-
- iii. Availability: Clients can access the system whenever an internet connection is accessible.
-
- iv. Recovery: In case of server failures or unavailability, the system should possess the capability to recover and manage any potential data loss or overflow.
-
- v. Flexibility and Portability: The system will offer accessibility from any location and at any time.

3.3 System Design

3.3.1 Design Constraints

Software Environment:

- Python Version: 3.12.1 or higher
- Libraries and Frameworks: Pandas, NumPy, scikit-learn, LightGBM, Streamlit
- Integrated Development Environment (IDE): Preferred IDE such as Jupyter Notebook, PyCharm, or VSCode.

Hardware Environment:

- Processor: Intel Core i7 or ryzen 7
- RAM: Minimum 8 GB RAM (16 GB recommended for large datasets and model training)
- Storage: Sufficient disk space for storing datasets, models, and application files
- Operating System: Compatible with Windows, macOS, or Linux distributions.

Experimental Setup:

- Access to the Spotify API for retrieving song data and features
- Utilization of a machine learning model training environment, possibly leveraging GPU acceleration for faster training (optional)
- Internet connectivity for accessing external datasets, APIs, and deploying the Streamlit application.

Data Constraints:

- Availability of labeled song dataset with emotions (happiness, sadness, calmness, energy)
- Quality and diversity of training data sourced from Spotify playlists and user-curated selections
- Compliance with data privacy regulations and ethical considerations regarding the use of user data.

Model Constraints:

- Use of LightGBM model for emotion recognition based on song features
- Interpretability and explainability of the model for understanding emotion inference
- Optimization of model hyperparameters for improved performance.

Deployment Constraints:

- Development of a Streamlit web application for deploying the MoodSonic recommendation system
- Compatibility with various web browsers and operating systems
- Integration with cloud hosting services (e.g., Heroku, AWS, or Google Cloud Platform) for deploying the web application.

Performance Constraints:

- Low latency and fast response times for generating recommendations
- Optimization of model inference and recommendation algorithms for efficient real-time performance
- Scalability to handle large user loads and datasets.

User Interface Constraints:

- Design of an intuitive and user-friendly interface for interacting with the recommendation system
- Responsive design for accessibility across different devices and screen sizes
- Incorporation of feedback mechanisms for users to provide input on recommendations.

Maintenance and Support Constraints:

- Implementation of version control and documentation practices for codebase maintenance
- Regular updates and bug fixes to address issues and improve functionality
- Availability of user support channels for troubleshooting and assistance with system usage.

Chapter 4

Implementation

In this section, present the implementation done by you during the project development.

1.1 Methodology OR Proposal

The MoodSonic project likely follows a Machine Learning (ML) approach for music emotion recognition and recommendation. Here's a breakdown of the potential methodological steps:

1. Data Collection and Pre processing:

- Wisdom of the Crowds: Leverage labeled data from user playlists and Spotify's curated selections for emotions like happiness, sadness, calmness, and energy.
- Data Balancing: Employ techniques to address potential imbalances in emotional representation within the dataset.

2. Feature Engineering:

Utilize the provided audio features from Spotify API:
- Acousticness
- Danceability
- Energy
- Instrumentalness
- Liveness
- Loudness
- Speechiness
- Tempo
- Valence
- Extract additional features if needed based on the chosen ML model.

3. Model Building and Training:

- Light Gradient Boosting Machine (LGBM) is mentioned as the candidate model.
- Train the LGBM model on the labeled dataset to predict the emotions associated with songs based on their audio features.
- Hyperparameter tuning can be performed to optimize the model's performance.

4. Recommendation System:

- Utilize Cosine Similarity: This mathematical approach can identify songs with similar audio feature profiles to a target song.
- Leverage the trained emotion recognition model: Recommend songs based on the predicted emotions and user preferences.

5. Deployment:

- Create a Streamlit application: This web framework facilitates the development of intuitive interfaces for engaging with the MoodSonic system. Users have the ability to input songs or artists and obtain suggestions tailored to their emotional inclinations.

4.2 Testing OR Verification Plan

To ensure MoodSonic's effectiveness in music emotion recognition and recommendation, a multi-faceted testing approach is recommended:

4.2.1 Data Testing:

- Data Quality Checks: Verify the dataset for completeness, consistency, and absence of errors.
- Exploratory Data Analysis (EDA): Analyze data distributions, identify outliers, and explore relationships between features and emotions.
- Data Balancing Evaluation: Assess the effectiveness of techniques used to balance the dataset during preparation.

4.2.2 Model Testing

- Training-Validation Split: Divide the dataset into training and validation sets. Train the model on the training set and evaluate its performance on the unseen validation set to prevent overfitting.
- Evaluation Metrics:
- Classification Accuracy: Measure the percentage of songs where the predicted emotion matches the actual label.

- Precision and Recall:
- Precision: Ratio of correctly predicted positive cases (e.g., happy songs).
- Recall: Ratio of all actual positive cases identified by the model.
- F1-Score: Combines precision and recall into a single metric.
- Hyperparameter Tuning Evaluation: Compare the model's performance with different hyperparameter configurations to identify the optimal settings.

4.2.3 Recommendation System Testing

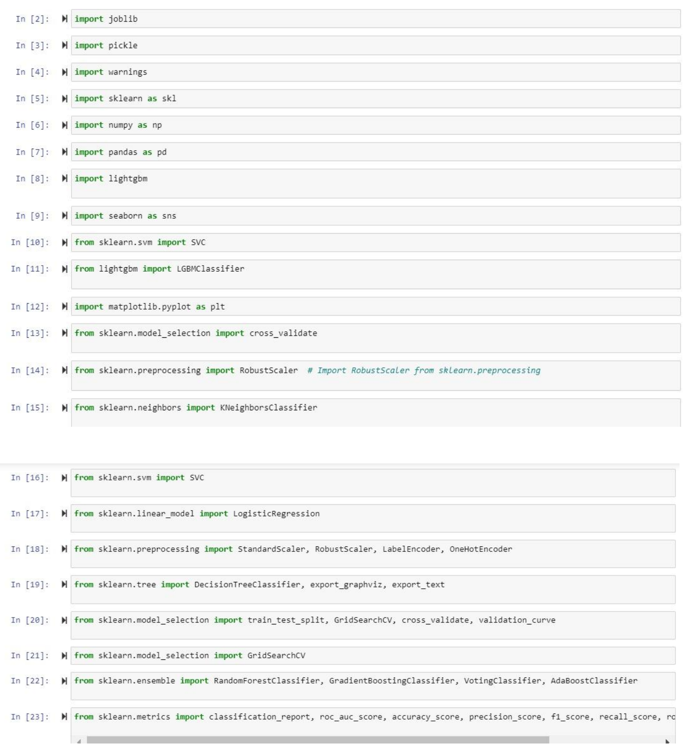
- User Evaluation: Conduct user studies where participants interact with the MoodSonic app and provide feedback on the accuracy and relevance of the recommendations. Consider A/B testing different recommendation strategies.
- Case Studies: Test the system with specific song examples or user profiles to verify if recommendations align with expected emotions.

4.2.4 Overall System Testing

- Functionality Testing: Verify that the Streamlit application functions as intended. Users can input songs, receive recommendations, and navigate the interface seamlessly.
- Performance Testing: Evaluate the application's response time and resource usage under various user loads.
- By implementing this testing plan, developers can gain confidence in MoodSonic's ability to accurately recognize song emotions and provide relevant music recommendations.

4.3 Result Analysis

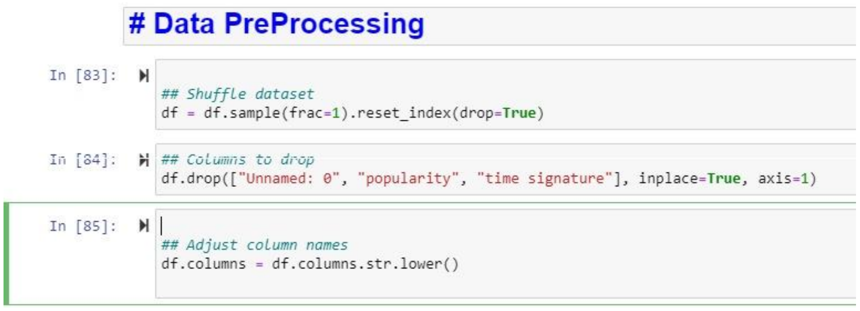
4.3.1 Importing libraries



```
In [2]:  import joblib
In [3]:  import pickle
In [4]:  import warnings
In [5]:  import sklearn as skl
In [6]:  import numpy as np
In [7]:  import pandas as pd
In [8]:  import lightgbm
In [9]:  import seaborn as sns
In [10]: from sklearn.svm import SVC
In [11]: from lightgbm import LGBMClassifier
In [12]: import matplotlib.pyplot as plt
In [13]: from sklearn.model_selection import cross_validate
In [14]: from sklearn.preprocessing import RobustScaler  # Import RobustScaler from sklearn.preprocessing
In [15]: from sklearn.neighbors import KNeighborsClassifier
In [16]: from sklearn.svm import SVC
In [17]: from sklearn.linear_model import LogisticRegression
In [18]: from sklearn.preprocessing import StandardScaler, RobustScaler, LabelEncoder, OneHotEncoder
In [19]: from sklearn.tree import DecisionTreeClassifier, export_graphviz, export_text
In [20]: from sklearn.model_selection import train_test_split, GridSearchCV, cross_validate, validation_curve
In [21]: from sklearn.model_selection import GridSearchCV
In [22]: from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifier, AdaBoostClassifier
In [23]: from sklearn.metrics import classification_report, roc_auc_score, accuracy_score, precision_score, f1_score, recall_score, rc
```

4.3.2 Data Preprocessing

Several data preprocessing steps are applied to prepare the data for model training.



# Data PreProcessing

```
In [83]:
## Shuffle dataset
df = df.sample(frac=1).reset_index(drop=True)

In [84]:
## Columns to drop
df.drop(["Unnamed: 0", "popularity", "time signature"], inplace=True, axis=1)

In [85]:
## Adjust column names
df.columns = df.columns.str.lower()
```

4.3.3 Model Building and Evaluation

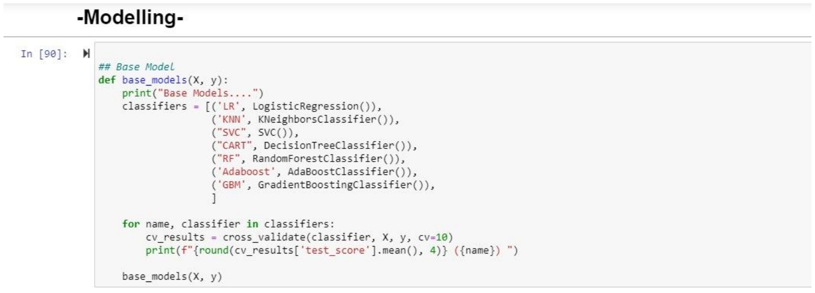
The data modelling stage focuses on building and evaluating machine learning models to predict song emotions based on the extracted features. The code implements two key approaches:

Baseline Models:

The base_models function trains and evaluates a variety of commonly used classification algorithms on the dataset (Section ## Base Model). These models include Logistic Regression (LR), K-Nearest Neighbors (KNN), Support Vector Machines (SVC), Decision Trees (CART), Random Forests (RF), AdaBoost, and Gradient Boosting Machines (GBM). Cross-validation is employed to assess the performance of each model, providing a more reliable estimate of their generalizability on unseen data. Evaluating these baseline models helps establish a benchmark for performance comparison and identify potential areas for improvement.

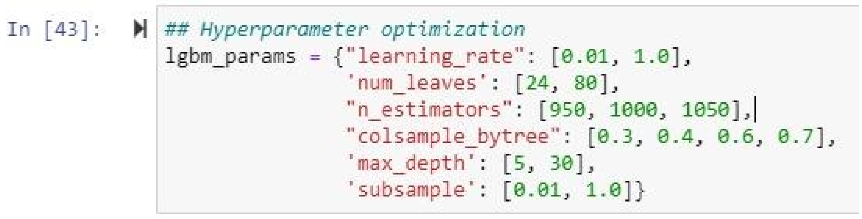
LightGBM with Hyperparameter Tuning:

LightGBM, a gradient boosting model known for its effectiveness in various classification tasks, is chosen for further investigation . Here, the code utilizes GridSearchCV to perform hyperparameter tuning. Hyperparameters are essentially the settings that control the learning process of a model. GridSearchCV systematically evaluates different combinations of hyperparameter values from a defined grid and selects the configuration that yields the best performance on the validation set. This process helps identify the optimal hyperparameter settings for LightGBM, potentially leading to improved emotion recognition accuracy.



-Modelling-

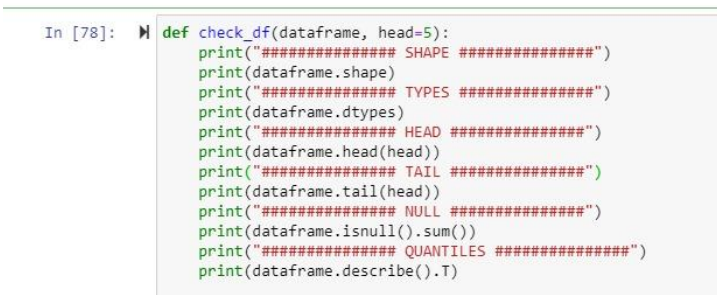
```
In [90]:
## Base Model
def base_models(X, y):
    print("Base Models....")
    classifiers = [('LR', LogisticRegression()),
                   ('KNN', KNeighborsClassifier()),
                   ("SVC", SVC()),
                   ("CART", DecisionTreeClassifier()),
                   ("RF", RandomForestClassifier()),
                   ('Adaboost', AdaBoostClassifier()),
                   ('GBM', GradientBoostingClassifier()),
                   ]

    for name, classifier in classifiers:
        cv_results = cross_validate(classifier, X, y, cv=10)
        print(f"{round(cv_results['test_score'].mean(), 4)} ({name}) ")

    base_models(X, y)
```



```
In [43]:  ## Hyperparameter optimization
          lgbm_params = {"learning_rate": [0.01, 1.0],
                         'num_leaves': [24, 80],
                         "n_estimators": [950, 1000, 1050],
                         "colsample_bytree": [0.3, 0.4, 0.6, 0.7],
                         'max_depth': [5, 30],
                         'subsample': [0.01, 1.0]}
```

4.3.4 Data Exploration

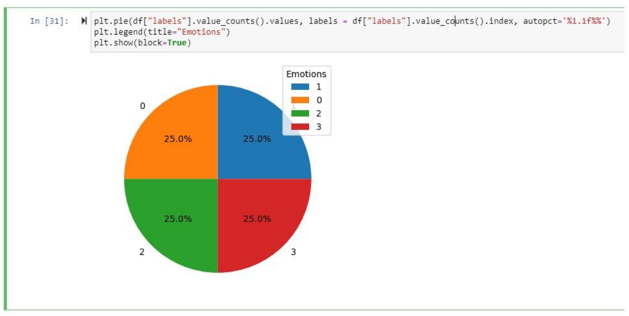
The initial data exploration phase involved reading the song data from a CSV file using pandas. A custom function check_df was used to analyze the data's shape, data types, presence of missing values, and basic descriptive statistics for numerical features. This initial assessment helps understand the data structure and identify potential issues before proceeding with further processing and model training.



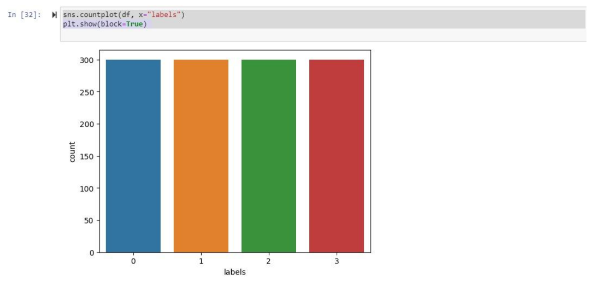
```
In [78]:  def check_df(dataframe, head=5):
              print("############### SHAPE ###############")
              print(dataframe.shape)
              print("############### TYPES ###############")
              print(dataframe.dtypes)
              print("############### HEAD ###############")
              print(dataframe.head(head))
              print("############### TAIL ###############")
              print(dataframe.tail(head))
              print("############### NULL ###############")
              print(dataframe.isnull().sum())
              print("############### QUANTILES ###############")
              print(dataframe.describe().T)
```

4.3.5 Data Visualization

To visualize the distribution of emotions in the song dataset , both a bar graph and a pie chart were generated. The bar graph displays the emotion counts (happiness, sadness, calmness, and energy) as vertical bars, while the pie chart shows the same information as proportional slices of a circle. This dual approach provides complementary insights: the bar graph offers a clear view of individual emotion frequencies, and the pie chart highlights the relative proportion of each emotion within the dataset. A balanced distribution across all emotions in these visualizations is desirable for training a robust music recommendation system.



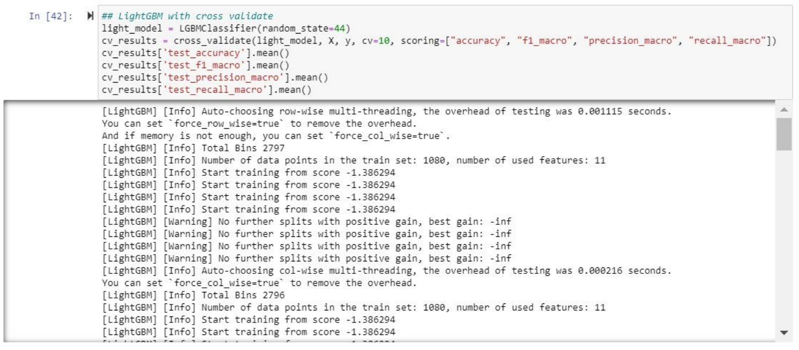
```
In [31]:  plt.pie(df["labels"].value_counts().values, labels = df["labels"].value_counts().index, autopct='%1.1f%%')
          plt.legend(title="Emotions")
          plt.show(block=True)
```



```
In [32]:  sns.countplot(df, x="labels")
          plt.show(block=True)
```

4.4.6 Classification

key metric calculated during this process is test accuracy, which reflects the average proportion of songs where the model correctly predicts the emotion on unseen data. This accuracy serves as a vital indicator of how well the LightGBM model can translate its learning to real-world scenarios involving new songs. We will explore additional performance metrics in the results section.



```
In [42]:  ## LightGBM with cross validate
          light_model = LGBMClassifier(random_state=44)
          cv_results = cross_validate(light_model, X, y, cv=10, scoring=["accuracy", "f1_macro", "precision_macro", "recall_macro"])
          cv_results['test_accuracy'].mean()
          cv_results['test_f1_macro'].mean()
          cv_results['test_precision_macro'].mean()
          cv_results['test_recall_macro'].mean()
```

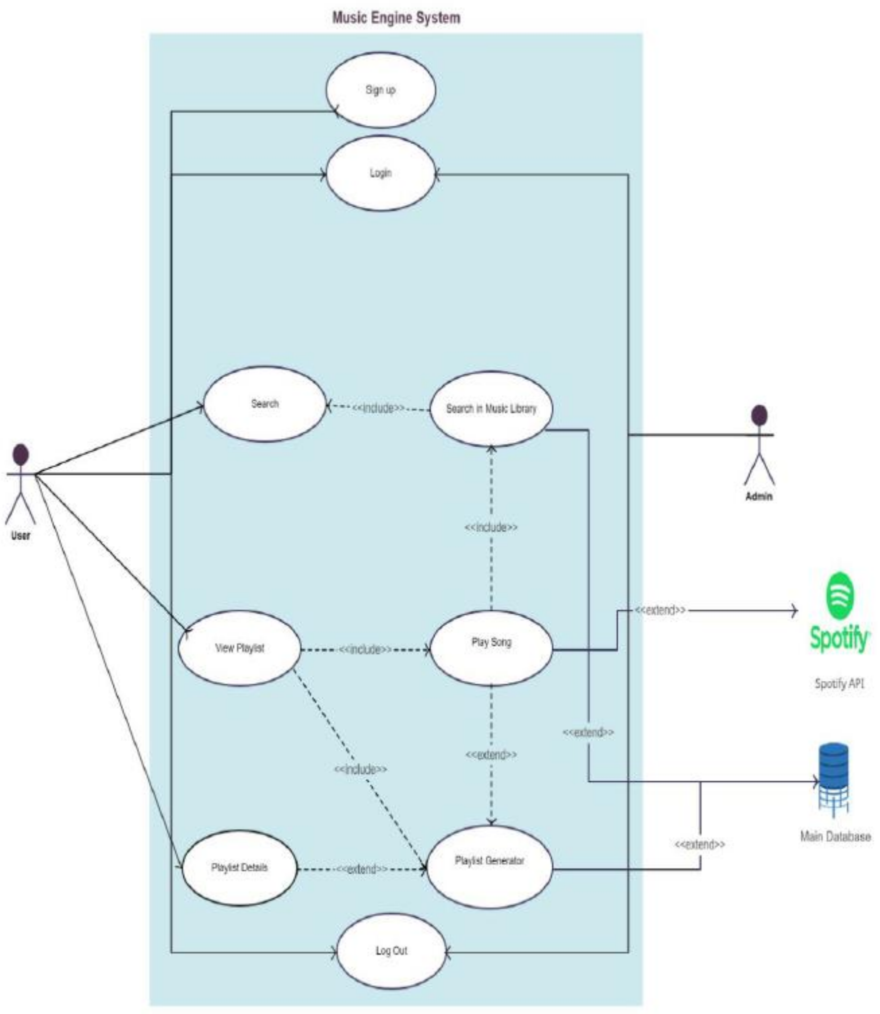
```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001115 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 2797
[LightGBM] [Info] Number of data points in the train set: 1080, number of used features: 11
[LightGBM] [Info] Start training from score -1.386294
[LightGBM] [Info] Start training from score -1.386294
[LightGBM] [Info] Start training from score -1.386294
[LightGBM] [Info] Start training from score -1.386294
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.000216 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 2796
[LightGBM] [Info] Number of data points in the train set: 1080, number of used features: 11
[LightGBM] [Info] Start training from score -1.386294
[LightGBM] [Info] Start training from score -1.386294
```

Chapter 5

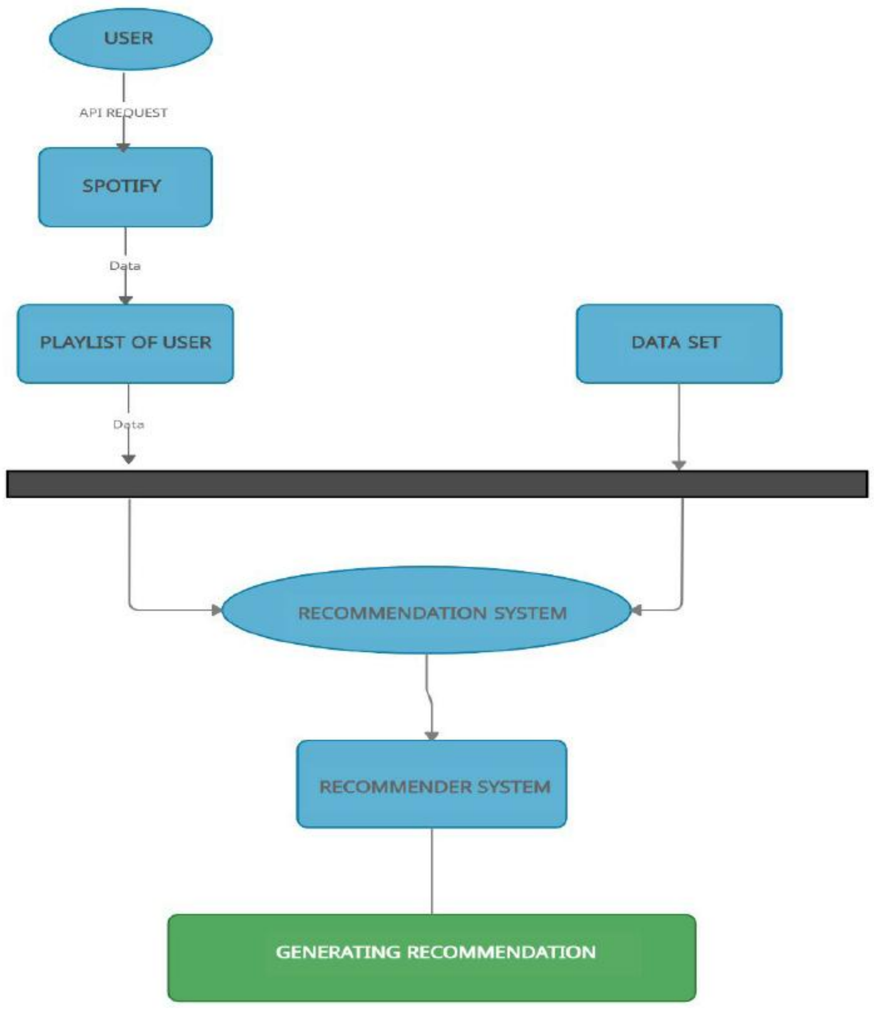
Standards Adopted

5.1 Design Standards

Use Case Diagram of the Project:



DFD Diagram of the Project:



5.2 Coding Standards

1. Consistent Naming : Use consistent and descriptive variable names throughout the code. For example, use df for data frames, X for feature matrix, y for target variable, etc.

2. Modularization : Break down the code into modular functions or sections, each serving a specific purpose. For example, separate data preprocessing, model building, evaluation, and visualization into distinct sections.

3. Comments and Documentation : Use comments to explain the purpose of each function, block of code, or important variables. Additionally, provide docstrings for functions to describe their inputs, outputs, and functionality.

4. Whitespace and Formatting : Use consistent whitespace and formatting conventions to improve code readability. For example, use consistent indentation (e.g., four spaces) and spacing around operators.

5. Avoid Magic Numbers and Strings : Avoid using magic numbers or strings directly in the code. Instead, define them as constants with descriptive names. For example, instead of 10 for the number of cross-validation folds, use NUM_FOLDS = 10.

6. Error Handling : Include appropriate error handling mechanisms to handle exceptions gracefully and provide informative error messages to users.

7. Avoid Hardcoding : Avoid hardcoding file paths, hyperparameters, or any other values directly into the code. Instead, use parameters or configuration files to make the code more flexible and reusable.

8. Consistent Imports : Import libraries in a consistent and organized manner. For example, group standard library imports together, followed by third-party library imports, and finally local imports.

9. Use of Constants : Use constants for values that are not expected to change during execution. This makes the code more readable and easier to maintain.

10. Consistent Style : Follow a consistent coding style throughout the codebase. If working in a team, agree upon a coding style guide and adhere to it consistently.

5.3 Testing Standards

For testing and verification of the project work, the following standards based on ISO and IEEE guidelines are recommended:

1. ISO/IEC/IEEE 29119 : This standard provides guidelines for the execution of software testing. It covers test processes, test documentation, test techniques, and test management.

2. ISO/IEC/IEEE 12207 : This standard defines the life cycle processes for software, including testing and quality assurance. It outlines the activities involved in testing and verification throughout the software development life cycle.

3. IEEE 829 : This standard specifies the format and content of software test documentation. It includes templates for test plans, test cases, test procedures, and test reports, ensuring consistency and completeness in testing documentation.

4. IEEE 610.12 : This standard defines the terminology used in software engineering, including terms related to testing and verification. It provides a common vocabulary for discussing testing activities and concepts.

5. ISO/IEC/IEEE 25010 : This standard outlines quality models and metrics for software product quality. It includes criteria for evaluating the effectiveness, efficiency, and satisfaction of software testing and verification activities.

6. ISO/IEC/IEEE 15288 : This standard defines the system life cycle processes, including testing and verification activities. It emphasizes the importance of integrating testing and verification activities with other system engineering processes.

7. IEEE 1028 : This standard provides guidelines for conducting reviews during the software development life cycle, including testing and verification reviews. It outlines roles, responsibilities, and procedures for conducting effective reviews.

Following these standards ensures that testing and verification activities are conducted systematically, documented appropriately, and integrated seamlessly into the software development life cycle. It helps in improving the quality, reliability, and maintainability of the software product.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

MoodSonic presents a promising approach to music recommendations that caters to users' emotional states. By leveraging machine learning and user-labeled emotional data, the project aims to transcend traditional recommendation systems that solely focus on genre, popularity, or listening history. Through the LGBM model and various audio features, MoodSonic strives to predict a song's emotional landscape and curate recommendations that resonate with users' emotional needs.

This document has outlined the potential design considerations, methodologies, testing strategies, and quality assurance practices that could be followed during the project's development. By implementing these methods, the MoodSonic team can build a robust and reliable system for personalized music recommendations.

6.2 Future Scope

The potential of MoodSonic extends beyond its initial implementation. Here are some exciting avenues for future exploration:

Enhancing Emotion Recognition:

Incorporate Additional Audio Features: The current approach utilizes features provided by the Spotify API. Exploring more advanced audio features extracted through specialized tools or deep learning techniques could potentially improve emotion recognition accuracy.
Multilingual Support: Expanding the project to accommodate music in different languages would require training models on multilingual datasets with appropriate emotional labels. This would broaden MoodSonic's reach to a global audience.
Explainable AI: Integrating techniques like LIME (Local Interpretable Model-agnostic Explanations) can help understand why the model makes specific emotion predictions. This transparency would increase user trust and allow for further refinement of the model based on user feedback.
Refining the Recommendation System:

Advanced Recommendation Algorithms: Implementing more sophisticated algorithms that consider user listening history, genre preferences, and time of day can lead to even more personalized recommendations. Collaborative filtering techniques that leverage the preferences of similar users could further enhance accuracy.
Mood-Specific Playlists: MoodSonic could move beyond individual song recommendations and curate playlists tailored to specific moods or activities (relaxation, focus, workout). This would provide a more holistic listening experience for users.
Real-Time Emotion Detection: Integrating real-time emotion detection through physiological sensors (e.g., heart rate, skin conductance) could personalize recommendations based on a user's current emotional state. This would create a dynamic and adaptive music experience.
Expanding User Interaction:

Mobile App Development: Developing a mobile application would allow users to access MoodSonic's functionalities on the go, further increasing its accessibility and convenience.
Social Integration: Integrating social features like sharing recommendations or creating collaborative playlists could foster a sense of community and shared musical discovery.
Mood Feedback Loop: Allowing users to provide feedback on the emotional accuracy of recommendations can create a closed-loop system for continuously improving the model's performance.
By pursuing these future directions, the MoodSonic project has the potential to evolve into a powerful tool for personalized music recommendations. It can transform music listening from a passive experience into a dynamic and emotionally-driven journey, catering to the ever-evolving needs and moods of its users. As the field of machine learning and music information retrieval continues to advance, MoodSonic can serve as a valuable platform for further exploration and innovation in the realm of music recommendation systems.

References:

1. Song, Y., Dixon, S., & Pearce, M. (2012, June). A survey of music recommendation systems and future perspectives. In 9th international symposium on computer music modeling and retrieval (Vol. 4, pp. 395-410).

2. Chen, H. C., & Chen, A. L. (2005). A music recommendation system based on music and user grouping. Journal of Intelligent Information Systems, 24, 113-132.

3. Chen, H. C., & Chen, A. L. (2001, October). A music recommendation system based on music data grouping and user interests. In Proceedings of the tenth international conference on Information and knowledge management (pp. 231-238).

4. Paul, D., & Kundu, S. (2020). A survey of music recommendation systems with a proposed music recommendation system. In Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018 (pp. 279-285). Springer Singapore.

5. http://ir.juit.ac.in:8080/jspui/bitstream/123456789/8276/1/Music%20Recommendation%20System.pdf

6. https://medium.com/codex/music-mood-classification-using-neural-networks-and-spotifys-web-api-d73b391044a4

7. https://www.eliftech.com/insights/all-you-need-to-know-about-a-music-recommendation-system-with-a-step-by-step-guide-to-creating-it/

8. https://www.kaggle.com/datasets/abdullahorzan/moodify-dataset/code

SAMPLE INDIVIDUAL CONTRIBUTION REPORT:

Music Recommendation System

JHANVI JAIN
21053233

Abstract: In response to the surge in online music consumption, our project introduces a specialized framework for personalized music recommendations. By analyzing user preferences and real-time data like heart rate, we aim to deliver tailored song suggestions via a mobile app integrated with smartwatches. Leveraging Machine Learning, our system categorizes songs based on various factors, enhancing user experience and navigating the vast music landscape efficiently.

Individual contribution and findings: My contribution to the project began with thorough research into existing research papers, where I explored various methodologies and techniques relevant to our project goals. Additionally, I extensively searched for datasets suitable for our project requirements, ensuring we had access to diverse and relevant data for analysis and model training. Furthermore, I contributed new ideas and approaches based on my findings, enriching our project's conceptual framework and enhancing its potential for innovation and impact.

Individual contribution to project report preparation: In preparing the group project report, my primary contribution was in providing content for the Introduction, Abstract, and Literature Review sections. I conducted extensive research to gather relevant information and insights, which were then synthesized and articulated to form cohesive and engaging introductory sections. Additionally, I contributed to crafting the abstract, ensuring it succinctly summarized the key aspects of the project. My role in the literature review involved reviewing existing research and synthesizing findings to provide context and background information for the project.

Individual contribution for project presentation and demonstration: Provided the contents for the PPT.

Full Signature of Supervisor:	Full signature of the student:
…………………………….	……………………………

SAMPLE INDIVIDUAL CONTRIBUTION REPORT:

Music Recommendation System

ANKITA KUAMRI
21053235

Abstract: With the digital music landscape evolving rapidly, our project introduces a sophisticated framework for music recommendations. By harnessing Machine Learning and real-time data from smartwatches, our mobile application delivers personalized song suggestions aligned with user preferences and context. Through categorizing songs based on various criteria, our system enriches the music discovery process, offering users a curated experience tailored to their tastes.

Individual contribution and findings: Leveraging my exploration of journals and reference sites, I contributed by proposing new features that could augment the functionality of our project. Through my research, I identified emerging trends and best practices in related fields, inspiring ideas for enhancements and extensions to our existing system. These ideas were rooted in insights gleaned from my review of academic literature and practical applications, enriching our project with fresh perspectives and innovative possibilities.

Individual contribution to project report preparation: In the final sections of the project report, I contributed to outlining the Standards Adopted, Conclusion, and Future Scope. I researched industry standards and best practices relevant to our project domain and documented them in the report. Additionally, I summarized the key findings and outcomes of our project in the Conclusion section, highlighting its significance and contributions. Furthermore, I provided insights and recommendations for future enhancements and extensions in the Future Scope section, drawing on our project experience and lessons learned.

Individual contribution for project presentation and demonstration: Described about Dataset, Feature Selection and Exploratory Data Analysis.

Full Signature of Supervisor: ……………………………

Full signature of the student: ……………………………..

SAMPLE INDIVIDUAL CONTRIBUTION REPORT:

Music Recommendation System

SOMYA SINHA
21053365

Abstract: Addressing the burgeoning demand for music recommendation systems, our project presents a novel framework focused on personalization and real-time data integration. By utilizing Machine Learning algorithms and smartwatch connectivity, our mobile application offers users tailored song suggestions based on factors like heart rate and time of day. Categorizing songs by genre, region, and more, our system enhances user satisfaction and streamlines music exploration.

Individual contribution and findings: My contribution to the project involved leveraging resources such as YouTube tutorials and dataset repositories to gain practical knowledge and insights. By referring to instructional videos and online resources, I gained valuable insights into implementation strategies and best practices for our project. Additionally, my exploration of dataset repositories allowed me to identify relevant datasets and understand their characteristics, facilitating informed decisions during the project development process.

Individual contribution to project report preparation: My contribution to the Implementation section of the project report involved detailing the technical aspects of our project implementation. I worked closely with the development team to document the various components, technologies, and methodologies employed in building the project. I provided detailed descriptions of the implementation process, including code snippets, algorithms, and workflow diagrams, to give readers a clear understanding of how the project was executed.

Individual contribution for project presentation and demonstration: Briefly described about Feature Engineering ,Hyperparameter Optimization and Conclusion.

Full Signature of Supervisor:	Full signature of the student:
…………………………….	……………………………..

SAMPLE INDIVIDUAL CONTRIBUTION REPORT:

Music Recommendation System

RAJBIR SINGH
21053357

Abstract: The proliferation of online music platforms necessitates advanced recommendation systems. Our project proposes a tailored framework aiming to deliver personalized song suggestions by incorporating real-time data like heart rate. Through Machine Learning techniques and integration with smartwatches, our mobile application offers users a curated music experience. By categorizing songs based on diverse criteria, our system enhances user engagement and simplifies music discovery.

Individual contribution and findings: In my role, I took the lead in implementing various aspects of the project, drawing on insights from research papers and academic literature to inform our approach. Additionally, I contributed ideas for incorporating different variables into our project, inspired by findings from relevant papers and research studies. By leveraging insights from academic sources, I guided the implementation process and proposed innovative features and functionalities to enhance the project's effectiveness and utility.

Individual contribution to project report preparation: I played a significant role in developing the Problem Statement and Requirement Specification sections of the project report. Drawing on insights from our project discussions and research findings, I outlined the primary challenges addressed by our project and defined the specific requirements and objectives. I collaborated closely with team members to ensure clarity and comprehensiveness in articulating the project's scope and goals, as well as specifying the functional and non-functional requirements.

Individual contribution for project presentation and demonstration: Gave the insight about the Introduction,System Architecture and Classifiers.

Full Signature of Supervisor: Full signature of the student:
……………………………. ……………………………..