# RHYTHMS OF THE MIND:
# MUSIC AS THERAPY FOR MENTAL WELL-BEING

A PROJECT REPORT

submitted by

**"Bhattacharya Brothers"**
(Soham Bhattacharya, M.Sc. Big Data Analytics, Semester 1, B2430059 &
Darpan Bhattacharya, M.Sc. Big Data Analytics, Semester 1, B2430044)

as a part of Machine Learning Course (CS230) of
Master of Science in Big Data Analytics program

**Department of Computer Science**
Ramakrishna Mission Vivekananda Educational and Research Institute
Belur, Howrah - 711202

24 November, 2024

# ACKNOWLEDGMENT

We would like to thank our course instructor, Br. Tamal mj, for providing us with the opportunity to work on such an interesting project. We also appreciate the freedom he offered in allowing us to choose our own topic for the final project of this course (CS230: Machine Learning).

Soham Bhattacharya (MSc BDA, Sem 1, B2430059)
Darpan Bhattacharya (MSc BDA, Sem 1, B2430044)

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# Chapter 1
# INTRODUCTION

Music is an art form that uses sound, rhythm, melody, and harmony to evoke emotions, tell stories, and communicate ideas. It has been a part of human culture for centuries, serving as both a form of expression and a means of connecting people across different cultures and generations. Music is diverse, spanning genres from classical to contemporary, and it plays a central role in entertainment, rituals, and personal experiences.

Mental health refers to a person's emotional, psychological, and social well-being, affecting how they think, feel, and act. It influences how individuals cope with stress, relate to others, and make choices. Good mental health is essential for overall well-being, while mental health disorders, such as anxiety, depression, and stress-related conditions, can have a significant impact on daily life. Mental health is influenced by a variety of factors, including genetics, life experiences, and environmental influences.

Music and mental health are deeply connected, as music has the power to influence emotions, reduce stress, and improve overall well-being. Research has shown that listening to or playing music can help alleviate symptoms of anxiety, depression, and trauma by stimulating the brain's reward system, lowering cortisol levels, and promoting relaxation. Music therapy, a structured form of treatment, is often used to support mental health recovery, offering a non-verbal way to express emotions, process experiences, and enhance cognitive functioning. Whether through soothing melodies or energizing rhythms, music plays a significant role in managing mental health and fostering emotional resilience.

The objective of our project is to analyze: **based on the music tastes and lifestyle of an individual, does music improve/worsen their mental health conditions**.

Our project uses a dataset which contains the survey of 736 individuals, and contains various self-reported information such as "age", "primary streaming service", "total time listened to music in a day", "favourite genre", "self-reported anxiety levels on a scale of 1-10", "self-reported insomnia level on a scale of 1-10", etc.

# Chapter 2
# LITERATURE REVIEW

Music Therapy is an exciting and growing area of research, and using machine learning techniques to assist music therapy has huge potential to make it more personalized and effective. By using ML, therapists can tailor music interventions based on a patient's emotional and physiological responses, which helps address a variety of needs, from mental health and stress relief to rehabilitation and cognitive improvement. ML techniques like classification, clustering, and recommender systems allow for customized music choices, while tools like emotion recognition and real-time biofeedback help adjust sessions in the moment. Although promising, there are still challenges to overcome, such as ensuring data privacy, making the models easier to understand, and integrating different types of data. With ongoing advancements, this fusion of technology and therapy has the potential to offer more individualized, adaptable care for patients.

We used a public dataset available here[1] consisting of various self-reported features such as age, hours of music listened to per day, genres of music listened to, various mental health features like anxiety, depression, etc. to analyze based on the music tastes and lifestyle of an individual, does music improve/worsen their mental health conditions.

To the best of our knowledge, although multiple works have been conducted using the dataset for Exploratory Data Analysis (EDA) purposes ([2], [3], [4]), no similar work has been done with the dataset for classification tasks.

# Chapter 3
# DATASET DESCRIPTION

For our project, we used a publicly available dataset [1] called "Music & Mental Health Survey Results", available on kaggle [5].
The dataset contains various attributes regarding an individual's music tastes and their mental health. Ultimately, there is a field called "music effects", which we use as a target variable to evaluate the influence of their music tastes and lifestyle on their mental health.

## ABOUT THE DATASET

The dataset is of shape $736 \times 33$, which implies that the raw, unprocessed data in our dataset consists of 33 features, and entries of 736 individuals. The features in our dataset are as follows:

- *Timestamp* : Date and time when form was submitted
- *Age* : Respondent's age
- *Primary streaming service* : Respondent's primary streaming service
- *Hours per day* : Number of hours the respondent listens to music per day
- *While working* : Does the respondent listen to music while studying/working?
- *Instrumentalist* : Does the respondent play an instrument regularly?
- *Composer* : Does the respondent compose music?
- *Fav genre* : Respondent's favorite or top genre
- *Exploratory* : Does the respondent actively explore new artists/genres?
- *Foreign languages* : Does the respondent regularly listen to music with lyrics in a language they are not fluent in?
- *BPM* : Beats per minute of favorite genre
- *Frequency [<Genre>]* : How frequently the respondent listens to that particular genre of music. There are 16 different fields for 16 different genres: Classical, Country, EDM, Folk, Gospel, Hip hop, Jazz, K pop, Latin, Lofi, Metal, Pop, R&B, Rap, Rock, and Video game music.

- *Anxiety* : Self-reported anxiety, on a scale of 0-10, higher the value, the more the anxiety
- *Depression* : Self-reported depression, on a scale of 0-10, higher the value, the more the depression
- *Insomnia* : Self-reported insomnia, on a scale of 0-10, higher the value, the more the insomnia
- *OCD* : Self-reported OCD, on a scale of 0-10, higher the value, the more the OCD
- *Music effects* : Does music improve/worsen respondent's mental health conditions? It can have 3 values: "improve", "no effect" and "worsen".
- *Permissions* : Permissions to publicize data

# Chapter 4
# DATA PREPROCESSING

## 4.1 DATA CLEANING

Firstly, we cleaned the data by droppping any entries with missing or null values in the dataset. We also temporarily added a column for preserving the original indices of the entries in the cleaned dataset. After cleaning the data, the dimension of our dataset reduced from $736 \times 33$ to $616 \times 34$.

## 4.2 FEATURE SELECTION

We then removed the features that wouldn't contribute towards our work, or would potentially impact the target variable insignificantly. We then calculated whether there were any entries that were outliers in our dataset, and then we dropped them.

## 4.3 FEATURE ENGINEERING

We performed label encoding to transform all categorical features into integers, which enabled them to be used for classification. We mapped the binary "Yes/No" features to $\{0, 1\}$, 1 denoting a "Yes" response and 0 denoting a "No" response. The *Frequency* features of various genres were mapped as {"Never": 0, "Sometimes": 1, "Rarely": 2, "Very frequently": 3}, and finally, the *Music Effects* feature was mapped as {"Worsen": 0, "No effect": 1, "Improve": 2}.

Now that we had a dataset that consisted of all integers, we now performed features scaling by normalizing the values for each feature so that the value for each feature for each entry was in the range $[0, 1]$.

So finally our dataset consisted of 616 entries, with each entry containing 23 numerical fields, each of them in the range $[0, 1]$.

### 4.4 TRAIN-TEST SPLIT

Finally we split our entire dataset into training and testing data. We allocated 70% of our dataset to our training data and the rest 30% to our testing data. We performed stratified sampling so that both healthy and unhealthy classes were represented proportionally in both sets.

# Chapter 5
# METHODOLOGY

## 5.1 LOGISTIC REGRESSION

### 5.1.1 What is it?

Logistic regression is a statistical technique used for binary classification, where the goal is to predict the probability of an outcome belonging to one of two classes (e.g., 0 or 1, Yes or No). Unlike linear regression, which predicts continuous values, logistic regression uses the logistic function (also known as the sigmoid function) to output probabilities that range between 0 and 1. The model works by taking a linear combination of input features, which are then passed through the sigmoid function to estimate the probability of the event occurring. This probability is used to classify the observation into one of the two categories: if the probability is greater than 0.5, it is classified as 1 (positive class), and if less, as 0 (negative class). Logistic regression is widely used in various fields, such as medicine, finance, and marketing, for tasks like disease prediction, fraud detection, and customer churn analysis. While it is simple, interpretable, and efficient, its major limitation is the assumption of a linear relationship between the independent variables and the log-odds of the dependent variable, which may not always hold in more complex data.

### 5.1.2 Justification

Since our problem is a classification problem, we used logistic regression as it performs classification. However, by default, logistic regression performs binary classification, but our problem consists of 3 classes. Thus, vanilla logistic regression would not be applicable to our problem. However, the logistic regression model offered by `scikit-learn`, which we used in our project, supports multi-class classification by using the One-v/s-Rest (OvR) strategy. Thus, we were able to apply logistic regression using the `lbfgs` solver in our project. We performed logistic regression both with and without hyperparameter tuning in our project.

## 5.2 SOFTMAX REGRESSION

### 5.2.1 What is it?

Softmax regression, also known as multinomial logistic regression, is an extension of logistic regression used for multi-class classification problems. It works by applying the softmax function to a vector of raw class scores (logits) to transform them into a probability distribution. This function ensures that the predicted probabilities for each class sum to 1. The model then selects the class with the highest probability as the predicted output. Softmax regression is commonly used in tasks where there are more than two possible categories, such as image classification and natural language processing.

### 5.2.2 Justification

We used softmax regression for the same reason that we used logistic regression. The primary difference between logistic and softmax regression is that softmax supports multi-class classification by default. In our project, we used softmax regression by setting the `multi_class` hyperparameter to `multinomial` in the `LogisticRegression` module. It uses a OvR strategy for multi-class classification.

## 5.3 RANDOM FOREST CLASSIFIER

### 5.3.1 What is it?

A Random Forest Classifier is an ensemble learning method used for classification tasks, which combines multiple decision trees to improve predictive accuracy and reduce overfitting. It works by constructing a collection of decision trees during training, where each tree is trained on a random subset of the data and features. The final prediction is made by aggregating the predictions from all the individual trees, typically using a majority vote for classification problems. This approach enhances the model's robustness, as it reduces the risk of relying on any single tree's bias or variance. Random Forest is widely used for its high accuracy, ability to handle large datasets, and resilience to overfitting, even with complex data.

### 5.3.2 Justification

We used Random Forest Classifier because it works well with both numerical and categorical data, can handle complex, non-linear relationships, and

is stable against overfitting because it is an ensemble method, which means that it combines multiple decision trees to create an overall strong model. A single decision tree might overfit the data or fail to capture its intricacies, but Random Forest Classifier overcomes that by aggregating predictions from multiple trees, potentially significantly improving accuracy and robustness. We used random forest classifier both with and without hyperparameter tuning in our project.

## 5.4 SUPPORT VECTOR MACHINE (SVM)

### 5.4.1 What is it?

Support Vector Machine (SVM) is a powerful supervised learning algorithm primarily used for classification tasks. It works by finding the hyperplane that best separates the data points of different classes in a high-dimensional feature space. SVM aims to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class, known as support vectors. This maximization leads to better generalization and reduces the risk of overfitting. SVM can also handle non-linear classification by applying kernel functions, which transform the data into higher-dimensional spaces where a linear separator can be found. SVM is effective in high-dimensional spaces and is widely used for tasks like text classification, image recognition, and bioinformatics.

### 5.4.2 Justification

We used SVM because it works well when the dataset has a rather large number of features relative to the number of samples, as in our case, where we have 616 samples for 23 features. It can also handle complex datasets with non-linear boundaries, and avoids overfitting by relying on the margin maximization principle. We used the Support Vector Classifier (`SVC`) from `scikit-learn`, which uses OvR strategy to perform multi-class classification. We used SVM both with and without hyperparameter tuning in our project.

## 5.5 XGBOOST

### 5.5.1 What is it?

XGBoost (Extreme Gradient Boosting) is an advanced machine learning algorithm that is part of the family of gradient boosting methods. Gradient

boosting is an ensemble technique where multiple weak models (typically decision trees) are trained sequentially. Each new model corrects the errors of the previous ones by focusing more on the misclassified instances.
XGBoost is particularly popular due to its high performance, speed, and accuracy. It's widely used for classification and regression tasks, and it often performs exceptionally well in competitive machine learning environments.

### 5.5.2 Justification

XGBoost is a very popular ensemble learning technique which is renowned for its speed and accuracy, especially with large and complex datasets. We used XGBoost because it is very fast and it often outperforms many other models in terms of predictive accuracy. It uses the OvR strategy to perform multi-class classification and is also very stable against overfitting. We used XGBoost both with and without hyperparameter tuning in our project.

## 5.6 LIGHTGBM

### 5.6.1 What is it?

LightGBM (Light Gradient Boosting Machine) is an open-source, distributed, high-performance implementation of the gradient boosting framework designed for speed and efficiency. It is widely used for classification, regression, and ranking tasks, especially when dealing with large datasets. LightGBM was developed by Microsoft and is optimized for both performance and memory efficiency.

### 5.6.2 Justification

We used LightGBM as yet another ensemble learning method for our project, primarily because of its high speed, flexibility and low memory usage. It is based on Gradient Boosting Decision Trees (GBDT), which is an ensemble method that builds a series of decision trees where each tree corrects the errors made by the previous one. LightGBM uses various advanced techniques to speed up training, which was useful for large dataset like the one used in this project, and it often outperforms other techniques in terms of accuracy with advanced techniques like leaf-wise growth and regularization.

## 5.7 VOTING CLASSIFIER

### 5.7.1 What is it?

A Voting Classifier is an ensemble machine learning model that combines the predictions of multiple individual models (also known as base models) to make a final decision. The idea is to improve the overall accuracy and robustness of the prediction by aggregating the outputs from several different models. It is particularly useful when you have multiple classifiers that perform well on different subsets of data or have different strengths in capturing patterns.
A Voting Classifier works by combining the predictions of several base classifiers and then using a voting mechanism to decide on the final output.
There are two primary types of voting:

1. **Hard Voting (Majority Voting)**: In hard voting, each individual model makes a prediction (e.g., class labels), and the final prediction is the class that receives the majority of votes.
   For example, if three classifiers predict the following classes for a given data point:
   Classifier 1: Class A
   Classifier 2: Class A
   Classifier 3: Class B
   The final prediction would be Class A, because it has received more votes (2 out of 3).
   Hard Voting is typically used for classification tasks when the individual classifiers are expected to output discrete class labels. The majority class label is chosen as the final prediction.

2. **Soft Voting (Probability Voting)**: In soft voting, each classifier predicts the probabilities of the different classes, and the final class is determined by averaging these probabilities.
   The class with the highest averaged probability across all classifiers is chosen as the final prediction.
   For example, if the classifiers output the following probabilities:
   Classifier 1: P(Class A) = 0.8, P(Class B) = 0.2
   Classifier 2: P(Class A) = 0.6, P(Class B) = 0.4
   Classifier 3: P(Class A) = 0.7, P(Class B) = 0.3
   The averaged probability for Class A would be $(0.8 + 0.6 + 0.7)/3 = 0.7$, and for Class B it would be $(0.2 + 0.4 + 0.3)/3 = 0.3$. Hence, the final prediction would be Class A.

Soft voting is used when the classifiers are capable of predicting probabilities (not just labels). This method is generally preferred when the classifiers provide accurate probability estimates, as it can give more nuanced predictions than hard voting.

### 5.7.2 Justification

We used Voting Classifiers because it would help us to compare between the different classification techniques and models that we used and would help us to combine all of their results and provide an overall better prediction on the dataset.
We used both hard and soft voting techniques in our project.

# Chapter 6
# IMPLEMENTATION

## 6.1 TOOLS AND LIBRARIES USED

We used the following tools for this project:

- We did our project using kaggle [5].
- We used the python [6] programming language for our project.
- We prepared our report and presentation using overleaf [7].
- Our project was uploaded on github [8] for reference.

We used the following python libraries in our project:

- `numpy` as `np`
- `pandas` as `pd`
- `matplotlib.pyplot` as `plt`
- `seaborn` as `sns`
- `time`, `warnings`, `os`
- `cross_val_score`, `train_test_split`, `GridSearchCV`, `RandomizedSearchCV` from `sklearn.model_selection`
- `plotly.express` as `px`
- `Counter` from `collections`
- `OrdinalEncoder`, `LabelEncoder`, `MinMaxScaler` from `sklearn.preprocessing`
- `LogisticRegression` from `sklearn.linear_model`
- `accuracy_score`, `confusion_matrix`, `classification_report`, `recall_score`, `precision_score`, `f1_score` from `sklearn.metrics`
- `RandomForestClassifier`, `VotingClassifier`, `AdaBoostClassifier`, `GradientBoostingClassifier` from `sklearn.ensemble`
- `SVC` from `sklearn.svm`
- `xgboost` as `xgb`
- `lightgbm` as `lgb`
- `stdev` from `statistics`

## 6.2 IMPORTANT HYPERPARAMETERS

The important hyperparamaters that we used in our project, and their values are as follows:

- *Logistic Regression*: `{'solver':'liblinear', 'penalty':'l2', 'l1_ratio':0.5, 'C': 0.01}`
- *Random Forest Classifier*: `{'n_estimators':200, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_depth': 20, 'bootstrap': True}`
- *SVM*: `{'C':0.1, 'degree':3, 'gamma':'scale', 'kernel':'linear', 'tol': 0.0001}`
- *XGBoost*: `{'colsample_bytree':0.8, 'gamma': 0, 'learning_rate':0.01, 'max_depth':3, 'n_estimators':200, 'subsample': 0.9}`

## 6.3 TRAINING PROCESS

We split our data into training sets and testing sets in a $70:30$ ratio. We performed stratified sampling so that both healthy and unhealthy classes were represented proportionally in both sets. We applied $k$-fold cross validation in the Voting classifiers, both hard and soft.

# Chapter 7
# RESULTS

## 7.1 CLASSIFICATION REPORTS

Table 7.1: Logistic Regression with Hyperparameter Tuning

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 4 |
| 1.0 | 0.00 | 0.00 | 0.00 | 41 |
| 2.0 | 0.76 | 1.00 | 0.86 | 140 |
| Accuracy | | | 0.76 | 185 |
| Macro avg | 0.25 | 0.33 | 0.29 | 185 |
| Weighted avg | 0.57 | 0.76 | 0.65 | 185 |

Table 7.2: Logistic Regression without Hyperparameter Tuning

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 4 |
| 1.0 | 0.57 | 0.10 | 0.17 | 41 |
| 2.0 | 0.78 | 0.99 | 0.87 | 140 |
| Accuracy | | | 0.77 | 185 |
| Macro avg | 0.45 | 0.36 | 0.34 | 185 |
| Weighted avg | 0.71 | 0.77 | 0.69 | 185 |

Table 7.3: Softmax Regression

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 4 |
| 1.0 | 0.57 | 0.10 | 0.17 | 41 |
| 2.0 | 0.78 | 0.99 | 0.87 | 140 |
| Accuracy | | | 0.77 | 185 |
| Macro avg | 0.45 | 0.36 | 0.34 | 185 |
| Weighted avg | 0.71 | 0.77 | 0.69 | 185 |

Table 7.4: Random Forest with Hyperparameter Tuning

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 4 |
| 1.0 | 0.00 | 0.00 | 0.00 | 41 |
| 2.0 | 0.76 | 0.99 | 0.86 | 140 |
| Accuracy | | | 0.75 | 185 |
| Macro avg | 0.25 | 0.33 | 0.29 | 185 |
| Weighted avg | 0.57 | 0.75 | 0.65 | 185 |

Table 7.5: Random Forest without Hyperparameter Tuning

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 4 |
| 1.0 | 0.12 | 0.02 | 0.04 | 41 |
| 2.0 | 0.76 | 0.96 | 0.85 | 140 |
| Accuracy | | | 0.74 | 185 |
| Macro avg | 0.30 | 0.33 | 0.30 | 185 |
| Weighted avg | 0.60 | 0.74 | 0.65 | 185 |

Table 7.6: SVM with Hyperparameter Tuning

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 4 |
| 1.0 | 0.00 | 0.00 | 0.00 | 41 |
| 2.0 | 0.76 | 1.00 | 0.86 | 140 |
| Accuracy | | | 0.76 | 185 |
| Macro avg | 0.25 | 0.33 | 0.29 | 185 |
| Weighted avg | 0.57 | 0.76 | 0.65 | 185 |

Table 7.7: SVM without Hyperparameter Tuning

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 4 |
| 1.0 | 0.57 | 0.10 | 0.17 | 41 |
| 2.0 | 0.78 | 0.99 | 0.87 | 140 |

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Accuracy | | | 0.77 | 185 |
| Macro avg | 0.45 | 0.36 | 0.34 | 185 |
| Weighted avg | 0.71 | 0.77 | 0.69 | 185 |

Table 7.8: XGBoost without Hyperparameter Tuning

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 4 |
| 1.0 | 0.25 | 0.05 | 0.08 | 41 |
| 2.0 | 0.77 | 0.97 | 0.86 | 140 |
| Accuracy | | | 0.75 | 185 |
| Macro avg | 0.34 | 0.34 | 0.31 | 185 |
| Weighted avg | 0.64 | 0.75 | 0.67 | 185 |

Table 7.9: XGBoost with Hyperparameter Tuning

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 4 |
| 1.0 | 0.25 | 0.05 | 0.08 | 41 |
| 2.0 | 0.77 | 0.97 | 0.86 | 140 |
| Accuracy | | | 0.75 | 185 |
| Macro avg | 0.34 | 0.34 | 0.31 | 185 |
| Weighted avg | 0.64 | 0.75 | 0.67 | 185 |

Table 7.10: LightGBM

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 4 |
| 1.0 | 0.38 | 0.12 | 0.19 | 41 |
| 2.0 | 0.78 | 0.96 | 0.86 | 140 |
| Accuracy | | | 0.75 | 185 |
| Macro avg | 0.39 | 0.36 | 0.35 | 185 |
| Weighted avg | 0.67 | 0.75 | 0.69 | 185 |

Table 7.11: Voting Classifier (Hard)

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 4 |
| 1.0 | 0.29 | 0.05 | 0.08 | 41 |
| 2.0 | 0.76 | 0.97 | 0.86 | 140 |
| Accuracy | | | 0.75 | 185 |
| Macro avg | 0.35 | 0.34 | 0.31 | 185 |
| Weighted avg | 0.64 | 0.75 | 0.67 | 185 |

Table 7.12: Voting Classifier (Soft)

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 4 |
| 1.0 | 0.14 | 0.02 | 0.04 | 41 |
| 2.0 | 0.76 | 0.97 | 0.86 | 140 |
| Accuracy | | | 0.74 | 185 |
| Macro avg | 0.30 | 0.33 | 0.30 | 185 |
| Weighted avg | 0.61 | 0.74 | 0.66 | 185 |

Table 7.13: Accuracy scores of all individually run models

| **Classifier** | **Training Accuracy Scores** | **Test Accuracy Scores** |
|---|---|---|
| Logistic Regression w hpt | 0.755814 | 0.756757 |
| Logistic Regression w/o hpt | 0.765116 | 0.767568 |
| Softmax Regression | 0.765116 | 0.767568 |
| RandomForest w hpt | 1.000000 | 0.756757 |
| RandomForest w/o hpt | 1.000000 | 0.735135 |
| SVM w hpt | 0.755814 | 0.756757 |
| SVM w/o hpt | 0.762791 | 0.756757 |
| XGBoost w/o hpt | 1.000000 | 0.745946 |
| XGBoost w hpt | 0.800000 | 0.735135 |
| LightGBM | 1.000000 | 0.751351 |
| Voting Classifier (Hard) | 0.981395 | 0.751351 |
| Voting Classifier (Soft) | 1.000000 | 0.740541 |

## 7.2 LINE GRAPH OF ACCURACY SCORES OF ALL MODELS

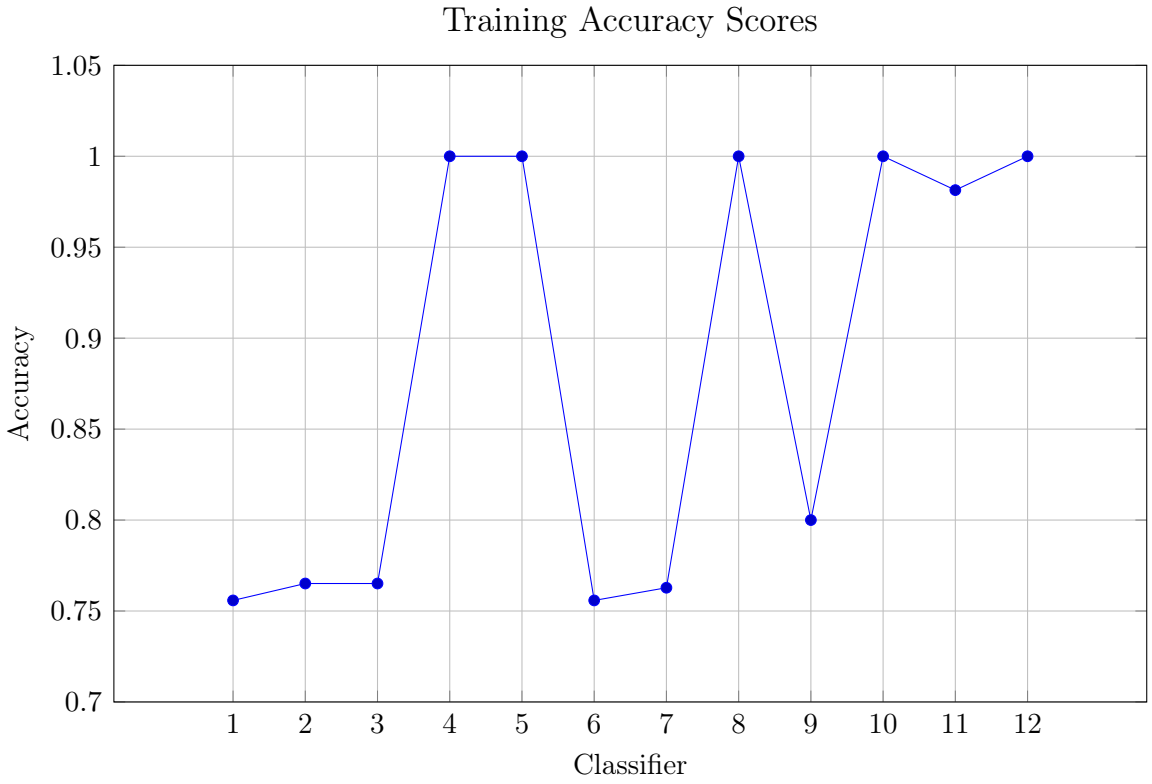


Figure 7.1: Training Accuracy Scores

Legend:
1 = Logistic Regression with hyperparameter tuning
2 = Logistic Regression without hyperparameter tuning
3 = Softmax Regression
4 = RandomForest with hyperparameter tuning
5 = RandomForest without hyperparameter tuning
6 = SVM with hyperparameter tuning
7 = SVM without hyperparameter tuning
8 = XGBoost without hyperparameter tuning
9 = XGBoost with hyperparameter tuning
10 = LightGBM
11 = Voting classifier with hard voting
12 = Voting classifer with soft voting

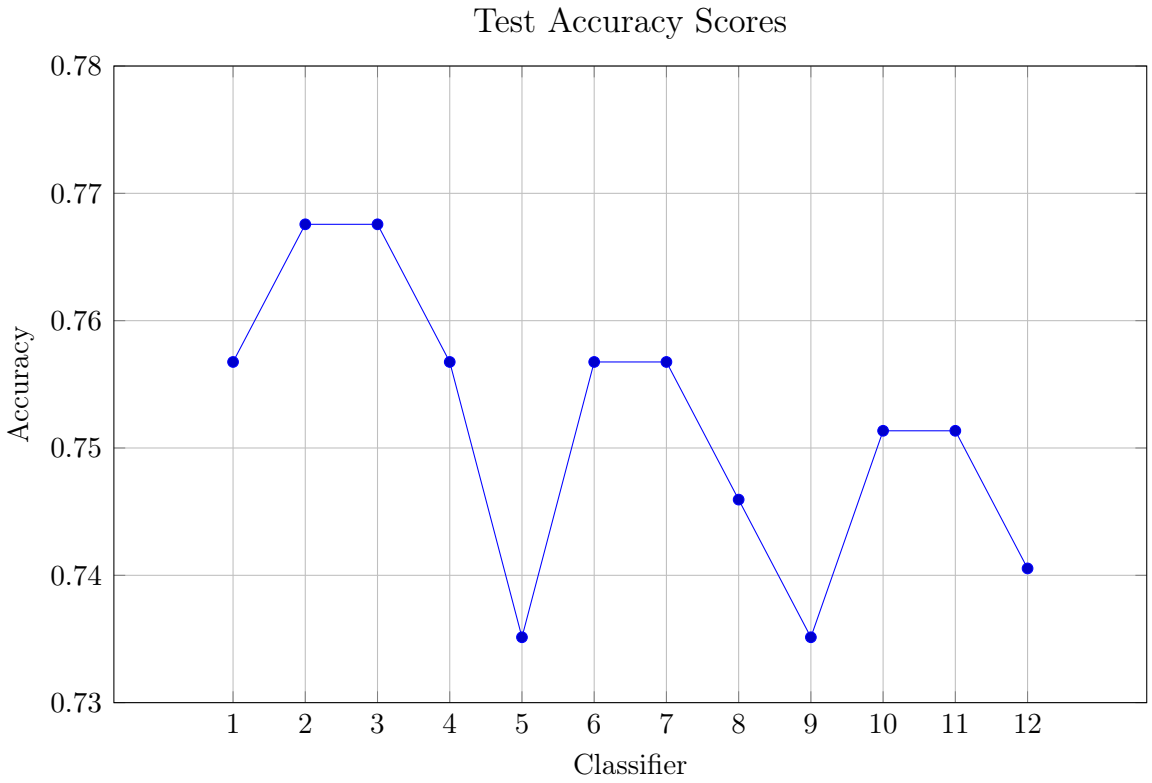


Figure 7.2: Test Accuracy Scores

Legend:
1 = Logistic Regression with hyperparameter tuning
2 = Logistic Regression without hyperparameter tuning
3 = Softmax Regression
4 = RandomForest with hyperparameter tuning
5 = RandomForest without hyperparameter tuning
6 = SVM with hyperparameter tuning
7 = SVM without hyperparameter tuning
8 = XGBoost without hyperparameter tuning
9 = XGBoost with hyperparameter tuning
10 = LightGBM
11 = Voting classifier with hard voting
12 = Voting classifer with soft voting

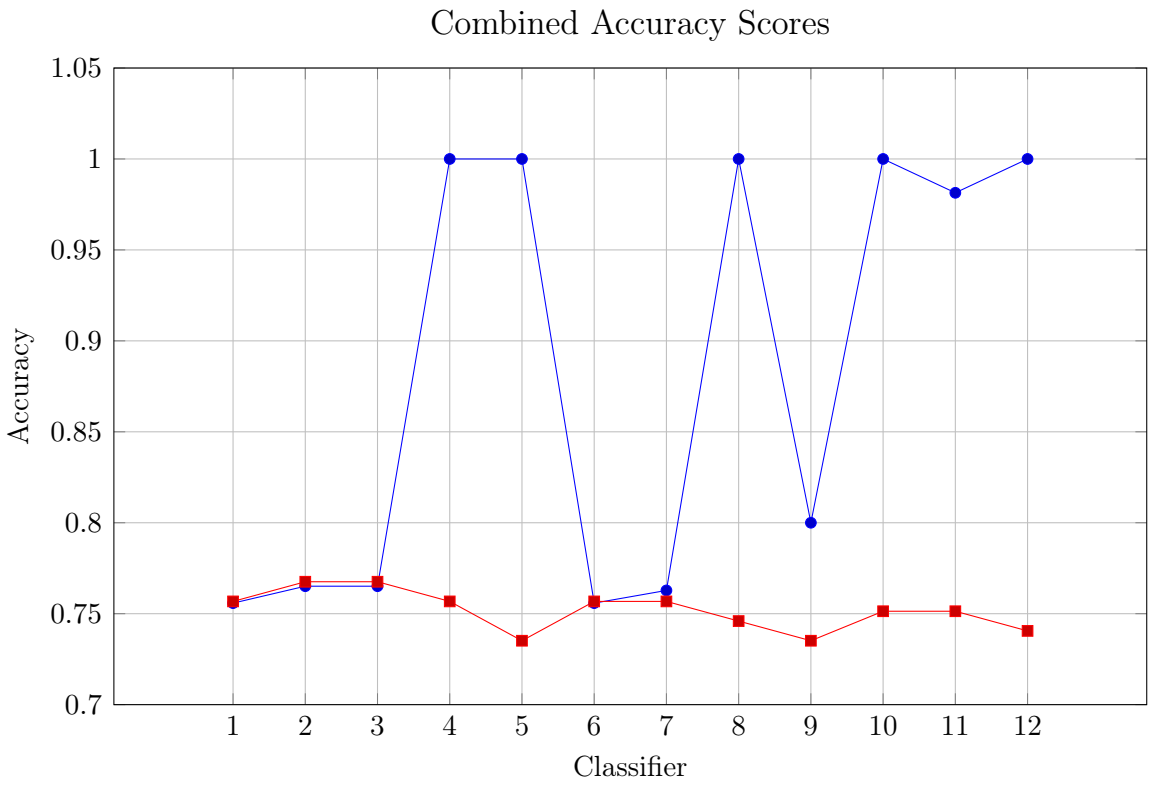


Figure 7.3: Combined Accuracy Scores (Training and Test)

Legend:
1 = Logistic Regression with hyperparameter tuning
2 = Logistic Regression without hyperparameter tuning
3 = Softmax Regression
4 = RandomForest with hyperparameter tuning
5 = RandomForest without hyperparameter tuning
6 = SVM with hyperparameter tuning
7 = SVM without hyperparameter tuning
8 = XGBoost without hyperparameter tuning
9 = XGBoost with hyperparameter tuning
10 = LightGBM
11 = Voting classifier with hard voting
12 = Voting classifer with soft voting

## 7.3 CORRELATION HEATMAP BETWEEN FEATURES AND TARGET VARIABLE

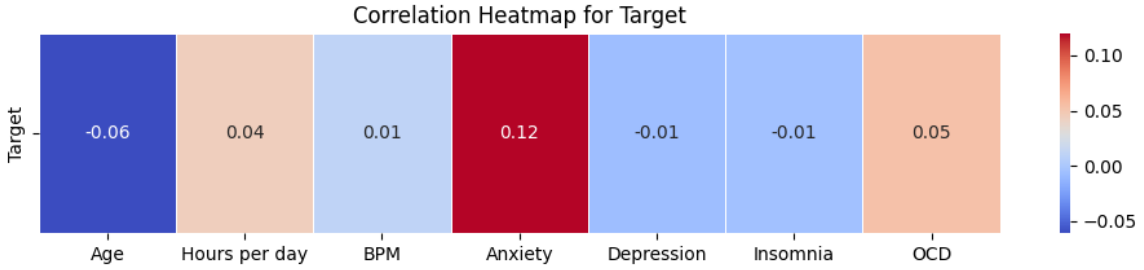


Figure 7.4: Correlation heatmap for non-frequency features

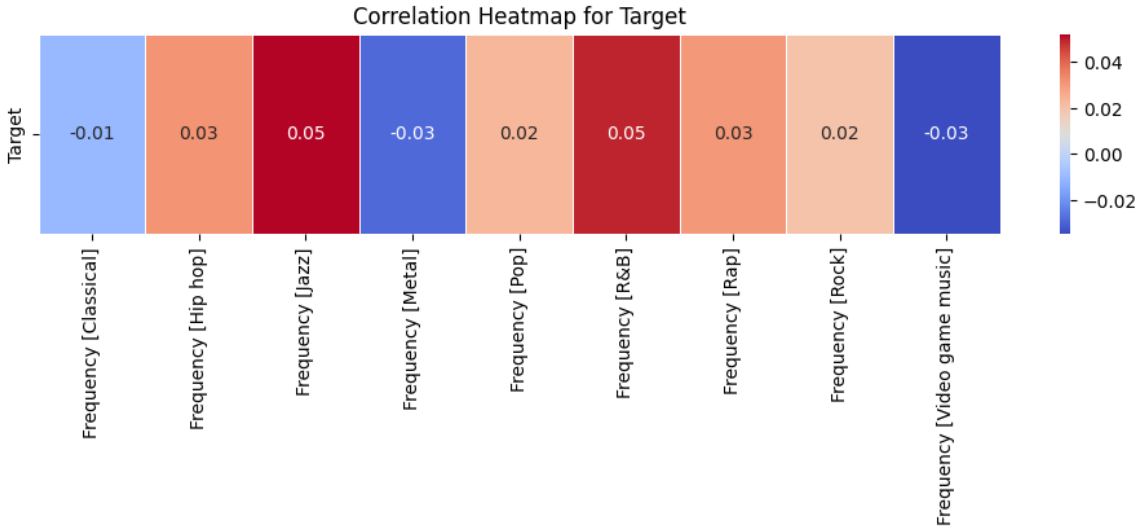


Figure 7.5: Correlation heatmap for frequency features

## 7.4 BEST RESULTS OBTAINED

From the various classification models that we trained our data on, we obtained the best prediction accuracy for the training set on **Logistic Regression without hyperparameter tuning** and **Softmax Regression**, where we obtained an accuracy of 0.767568. We also obtained very close prediction accuracies using **Logistic Regression with hyperparameter tuning** and **Random Forest with hyperparameter tuning**, where we obtained an accuracy of 0.756757.

Thus, to summarize:
**HIGHEST PREDICTION ACCURACY OBTAINED** = 0.767568
**MEAN PREDICTION ACCURACY OBTAINED** = 0.751801

# Chapter 8
# DISCUSSION & CONCLUSION

We obtained a prediction accuracy of $\sim 0.76$ from our project. Based on the analysis covered in our project our assumption that "given the music tastes and lifestyle of an individual, does music improve/worsen their mental health conditions" is indeed true. Our accuracy score depicts that we can detect an individual's mental health correctly with moderately high accuracy based on their living conditions and music tastes.
There were few limitations that we encountered during our study. Some of the more significant ones are:

- Due to resource constraints, we could not perform cross-validation for each and every model that we used in our project.
- The dataset that we had chosen for our project was not huge enough to offer very significant results. However we went ahead with it because it was the closest dataset that we found that both aligned with our interests and our objective.

## FUTURE SCOPE

In future, our project can be extended to further fine-tune the models to provide better results. Fine tuned models will better detect the mental health of the patient based on given lifestyle conditions and music tastes. Moreover, this aim will aid to better understand the present mental health condition of the patient and help in better diagnosis of the patient via music therapy.

# BIBLIOGRAPHY


[1] dataset. https://www.kaggle.com/datasets/catherinerasgaitis/mxmh-survey-results/data. [Accessed Nov 2024].

[2] Bashar Salman. Music and mental health - eda, 2024. Accessed: 2024-11-24.

[3] Melissa Monfared. Mental health and music relationship analysis - eda, 2024. Accessed: 2024-11-24.

[4] Yuna Sheng. Mental health and music relationship analysis - eda, 2024. Accessed: 2024-11-24.

[5] Kaggle. Kaggle: Your home for data science and machine learning, 2024. Accessed: 2024-11-24.

[6] Python Software Foundation. Python programming language, 2024. Accessed: 2024-11-24.

[7] Overleaf. Overleaf: Online latex editor, 2024. Accessed: 2024-11-24.

[8] Soham B. ML final project, 2024. Accessed: 2024-11-24.