Impact of Music on Mental Health: A Classification Analysis

Pablo Armando Uscanga Camacho
Universidad Panamericana

June 15, 2025

Contents

Chapter 1

Introduction

This report details a machine learning project focused on understanding the relationship between music listening habits and self-reported mental health indicators using the ‘mxmh_survey_results.csv’ dataset. The project’s primary objective is to build a classification model to predict the perceived effect of music on an individual’s mental health, categorizing it as ’Improve’, ’No effect’, or ’Worsen’. The report covers essential stages, including data exploration, preprocessing, and the theoretical foundations of various classification models. The aim is to identify key features influencing these effects and provide insights into the relationship between music habits and mental health. Evaluation metrics such as confusion matrices, accuracy, per-class precision, recall, and F1-score are utilized to assess model performance, especially given the potential for imbalanced classes.

Chapter 2

Project Definition

The objective of this project is to classify the impact of music therapy in a wide range of ages to improve mood, stress, and general mental health, by identifying the correlation between genres, listening time, and other variables. The purpose of this project is to predict the 'music effects' based on the individual's music listening habits and demographic information.

2.1 Project Expectations

- Develop a classification model to predict the effect of music on mental health ('Improve', 'No effect', 'Worsen').
- Identify key features that are the most influential in determining these effects.
- Provide insights into the relationship between music habits and mental health.

2.2 Metrics

For us to be capable of comparing the performance of the models, metrics are needed to evaluate them. In multiclass classification problems, where the task is to categorize instances into one of several classes, various evaluation metrics are used to assess the performance of the model. These metrics give different insights into how well the model performs, particularly when dealing with imbalanced classes. The metrics that will be used here will be the following:

- **Confusion matrix:** It's a table used to describe the performance of a classification model. It provides information on false positives, false negatives, true positives, and true negatives for each class.

Actual \ Predicted	**Class 1**	**Class 2**	**Class 3**
Class 1	TP_1	FN_1	FN_1
Class 2	FN_2	TP_2	FN_2
Class 3	FN_3	FN_3	TP_3

- **Accuracy:** It represents the proportion of total correct predictions (both positive and negative) out of all predictions made.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

- **Per-class metrics:** Crucial for understanding performance on each specific outcome ('Improve', 'No effect', 'Worsen'), especially if there is class imbalance.
  - **Precision:** The ratio of true positive predictions to total predicted positives (how many of the predicted instances for a given class were correct).

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i}$$

  - **Recall:** The ratio of true positive predictions to the total actual positives (how many of the actual instances for a given class were correctly identified).

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i}$$

  - **F1-score:** The harmonic mean of precision and recall. It gives a balance between precision and recall, especially when the class distribution is imbalanced (where one class might be more frequent than the others).

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

- **Cohen's kappa:** Statistical measure used to assess the agreement between two raters (or observers) when classifying items into categories.

Reliance solely on accuracy can be misleading, especially in scenarios where the classes within the target variable are imbalanced. For example, if one class constitutes the vast majority of the data points, a model could achieve a high overall accuracy by simply predicting the majority class for most instances while performing poorly on the minority class. In such cases, accuracy wouldn't truly reflect the model's ability to identify the less frequent, but potentially more critical, class.

A comprehensive evaluation therefore necessitates a deeper examination of precision, recall, and F1-score, which offer a more nuanced understanding of model performance across different classes. Being a per-class breakdown, these metrics reveal where the model excels and where it struggles. The selection of the most appropriate metric depends heavily on the specific problem's context and the relative costs associated with false positives and false negatives.

In multiclass or imbalanced tasks, a high accuracy with a low kappa value can reveal a model that is biased toward dominant classes. So, Cohen's Kappa is especially helpful when you have unequal class distribution, or when you're avoiding over-optimistic accuracy.

Considering the dataset's origin from self-reported survey results , it is plausible that it might exhibit a tendency to report positive or neutral experiences with music on mental health ('Improve', 'No effect') more frequently than negative ones ('Worsen'). This could lead to a significant class imbalance within the 'Music effects' target variable. Therefore, it becomes imperative to prioritize the evaluation of metrics such as Precision, Recall, and F1-score for each individual class, particularly for any minority classes.

Chapter 3

Data Exploration

In order to obtain the model, the *mxmh_ survey_ results.csv* dataset, encompassing 736 rows and 33 columns:

`‘Timestamp’, ‘Age’, ‘Primary streaming service’, ‘Hours per day’, ‘While working’, ‘Instrumentalist’, ‘Composer’, ‘Fav genre’, ‘Exploratory’, ‘Foreign languages’, ‘BPM’, ‘Frequency [Classical]’, ‘Frequency [Country]’, ‘Frequency [EDM]’, ‘Frequency [Folk]’, ‘Frequency [Gospel]’, ‘Frequency [Hip hop]’, ‘Frequency [Jazz]’, ‘Frequency [K pop]’, ‘Frequency [Latin]’, ‘Frequency [Lofi]’, ‘Frequency [Metal]’, ‘Frequency [Pop]’, ‘Frequency [R&B]’, ‘Frequency [Rap]’, ‘Frequency [Rock]’, ‘Frequency [Video game music]’, ‘Anxiety’, ‘Depression’, ‘Insomnia’, ‘OCD’, ‘Music effects’, ‘Permissions’`

`‘Permissions’` will be dropped for the classification, and all the `‘Frequency [Genre]’` shall be shortened in order to reduce redundancy.

3.1 Attributes

Describe the features (input attributes) that will be used to train the model and the target variable (output attribute) that the model will predict.

- **Timestamp**: Date and time when the form was submitted. Functions as ID.
- **Age**: Respondent’s age.
- **Hours per day**: Number of hours the respondent listens to music per day.

- **While working**: Whether the respondent listens to music while studying or working (binary: Yes/No).
- **Instrumentalist**: Whether the respondent plays an instrument regularly (binary: Yes/No).
- **Composer**: Whether the respondent composes music (binary: Yes/No).
- **Fav genre**: Respondent's favorite or top genre.
- **Exploratory**: Whether the respondent actively explores new artists/-genres (binary: Yes/No).
- **Foreign languages**: Whether the respondent regularly listens to music with lyrics in a language they are not fluent in (binary: Yes/No).
- **BPM**: Beats per minute of favorite genre.
- **Frequency [Genre]**: How frequently the respondent listens to that specific genre (Never, Rarely, Sometimes, Very frequently).
- **Anxiety**: Self-reported anxiety level (on a scale of 0 to 10).
- **Depression**: Self-reported depression level (on a scale of 0 to 10).
- **Insomnia**: Self-reported insomnia level (on a scale of 0 to 10).
- **OCD**: Self-reported OCD level (on a scale of 0 to 10).
- **Music effects**: The perceived effect of music on the respondent's mental health conditions (Improve, No effect, Worsen). This will be our target attribute.

3.2 Data Description

Having defined the attributes, we are now able to describe the dataset. This is done to obtain more information about the data set and determine if preprocessing is needed.

As seen in Table 3.1, we can observe that most of the columns in the dataset are objects, indicating a large number of categorical variables. We can also observe that most of the columns have a low or null number of missing values in them, except for BPM. As it is lower than 15%, we can impute (fill in the missing values) the column. In the last row, we can also observe some missing values in our target variable. The corresponding rows can be dropped, as they will not contribute to the classification model.

Table 3.1: Dataset Attributes Summary

Attribute	**Data Type**	**Missing Values (%)**
Age	float64	0.13
Primary streaming service	object	0.13
Hours per day	float64	0.0
While working	object	0.4
Instrumentalist	object	0.54
Composer	object	0.13
Fav genre	object	0.0
Exploratory	object	0.0
Foreign languages	object	0.0
BPM	float64	14.53
Frequency [Genres]	object	0.0
Anxiety	float64	0.0
Depression	float64	0.0
Insomnia	float64	0.0
OCD	float64	0.0
Music effects	object	1.08

3.3 Data Distribution

From these descriptions, we can classify the variables into numerical or categorical. These is done in order to identify which variables will need to undergo encoding or scaling for the model, due to the difference in interpretation from the model.
Numerical variables represent quantities or measurable amounts, and can be expressed directly as numbers (integers or decimals). Categorical variables represent categories, labels, or groups; these can be alphanumerical or represented by a number (e.g. 0 = No, 1 = Yes).

- **Numerical**: `‘Age’`, `‘Hours per day’`, `‘BPM’`, `‘Anxiety’`, `‘Depression’`, `‘Insomnia’`, `‘OCD’`
- **Categorical**: `‘Timestamp’`, `‘Primary streaming service’`, `‘While working’`, `‘Instrumentalist’`, `‘Composer’`, `‘Fav genre’`, `‘Exploratory’`, `‘Foreign language’`, `‘Frequency [Genre]’`, `‘Music effects’`

3.3.1 Numerical attributes

We can make use of an assorted number of graphs to gain more insight about the dataset's variables, such as understanding data distribution, identifying patterns, summarize the data, and identify outliers. For this dataset's numerical variables we'll use histograms, box plots, and density plots to attain a better understanding of our data.

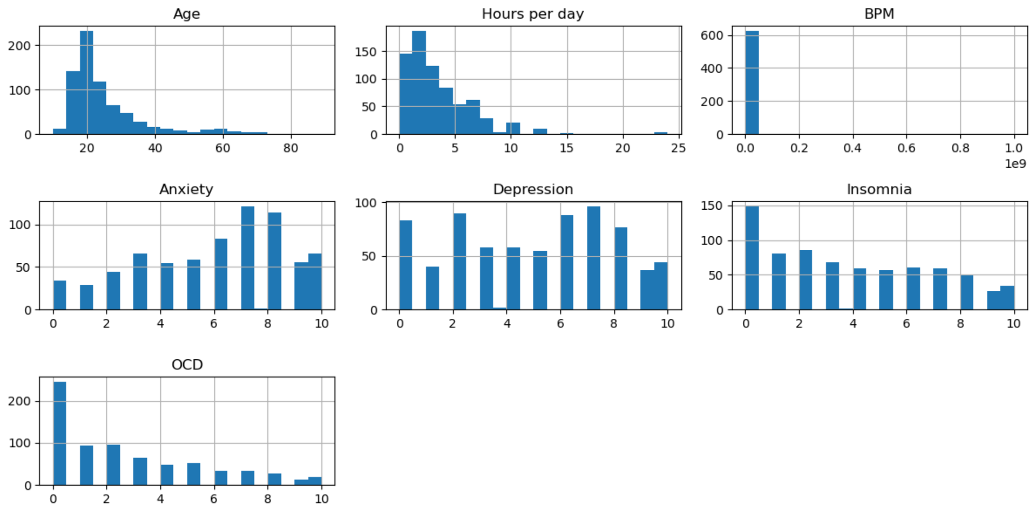


Figure 3.1: Histograms of Numerical Features

A histogram shows the distribution of the values by dividing the data into discrete intervals and counting how many data points fall into each value. From the histograms presented in Figure 3.2 we can begin to see some aspects about the variables, such as a possibility of a unique value or a grand outlier in the 'BPM' column, as well as the 'Age' and 'Hours per day' columns being positively skewed.

A box plot provides a graphical summary of the distribution of the variables by showing their minimum, first quartile (Q1), median, third quartile (Q3), and maximum. It also shows any outliers that may be present. The quartiles can be defined as:

- **First Quartile (Q1):** it's the median of the lower half of the dataset, it represents the value below which 25% of the data falls.
- **Median (Q2):** it's the middle value of the dataset, splitting the data into two equal halves.

- **Third Quartile (Q3):** it's the median of the upper half of the dataset, it represents the value below which 75% of the data falls.
- **Interquartile (IQR):** it's the difference between the third quartile and the first quartile, it tells how spread out the middle 50% of the data is spread out.

The box plots presented in Figure 3.2 clearly show us an outlier in the 'BPM' column, obstructing the view of the actual values in the column. We can also see some outliers in the 'Age' and 'Hours per day' columns, with a general tendency for lower values. As for the other variables, we can observe a good distribution , as well as a generally low level of OCD but a relatively high level of Anxiety on most testers.

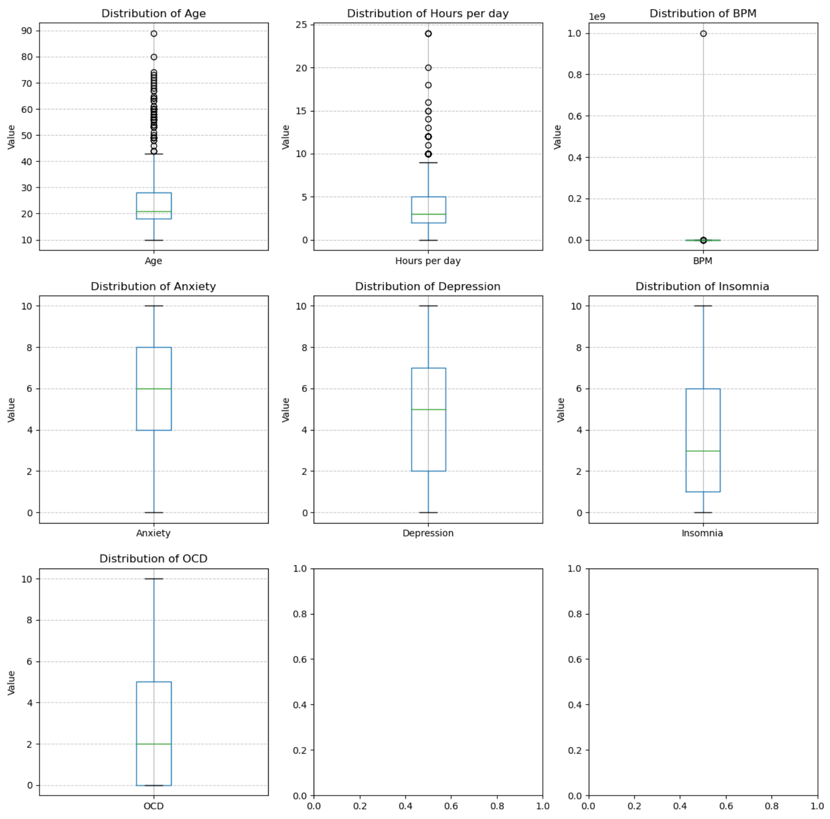


Figure 3.2: Box plots of Numerical Features

A density plot (also called kernel density estimate) is a smooth curve that represents the distribution of an individual attribute, telling us the shape of the distribution and density of the data. For the density plots[1] shown in Fig 3.3 we can reaffirm what we observed in the histograms, while also making new ones. ‘Age’, ‘Hours per day’, and ‘OCD’ are positively skewed, and the two peaks in ‘Depression’ (on 2 and 7) indicate a split between being relatively unaffected and moderately affected.

After analyzing these graphs, we can start to conclude that the dataset consists majorly of young testers, that listening hours are mostly low but with a few outliers (wether because of a data entry error or unusual habits), a generally high level of anxiety and depression, and OCD is less prevalent but still present.

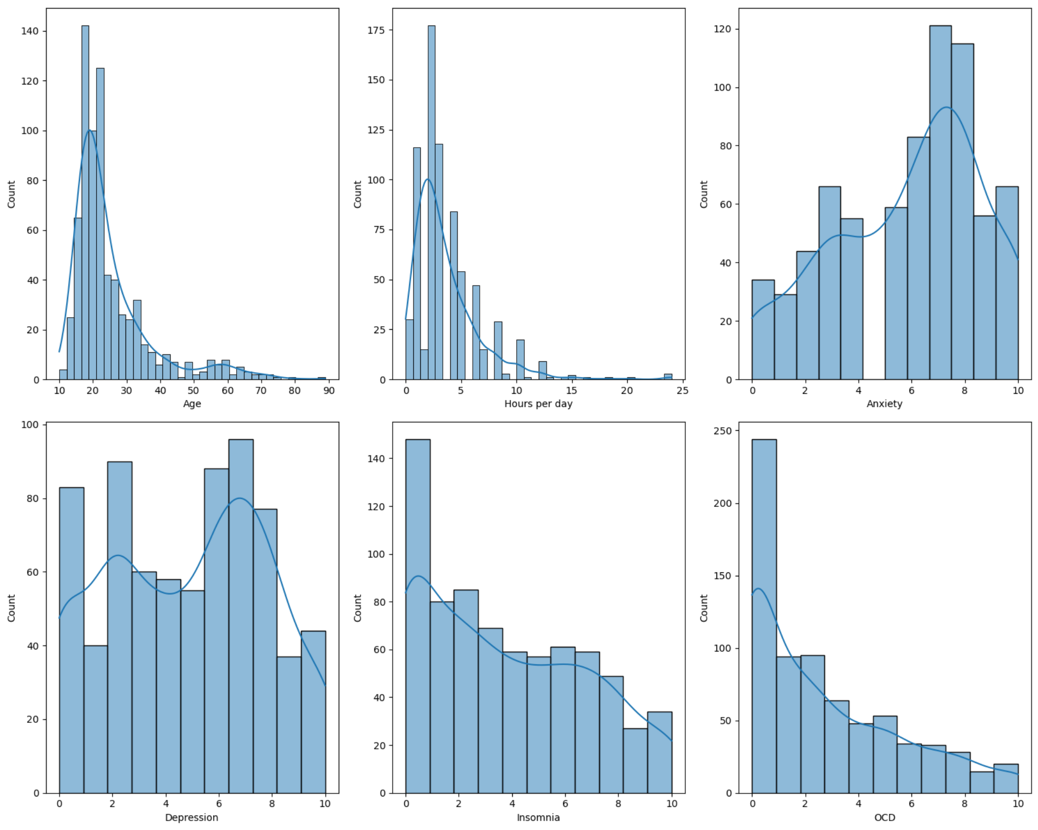


Figure 3.3: Density plots of Numerical Features

[1]The column ‘BPM’ is omitted due to a problem between seaborn (data visualization library) and that specific column, most likely due to the column’s dtype and/or outliers

3.3.2 Categorical attributes

Similar to the numerical, the categorical variables can make use of graphs to help identify patterns, make comparisons between categories, and understand the structure of the dataset.

For these features, we've split up the histograms into 3 figures for better comprehension. The first figure, Figure 3.4, helps us shed some light on the testers' characteristics. We can observe a grand number of non-composer listeners, doing so mainly while working, with some general openness to new music. As for our target variable, most of the results indicate an improvement on general mood, and a great lack of worsening. This lack of results in the worsen category will prove harmful to our classification models due to a lack of information for that class, so we will have to stratify our train, test, and validation sets.

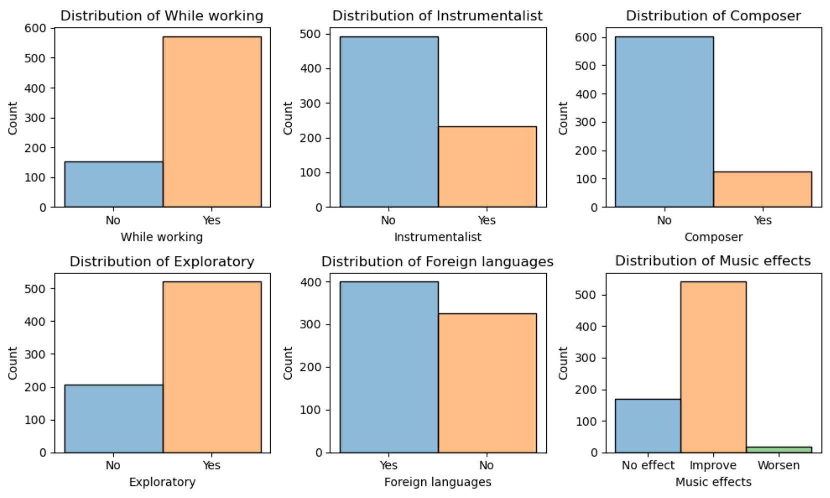


Figure 3.4: Main binary categorical attributes and target variable

In figure 3.5 we can observe Rock as the most preferred genre among most listeners, this being followed by Pop and Metal. Latin and Gospel appear to be the least favorite genre. Figure 3.6 shows us that many testers either have strong preferences or avoid certain genres entirely. The most popular genres are Pop, Rock, and Hip Hop, while Classical, Gospel, and Folk are rarely listened to.

3.4 Data Correlation

Finding the correlation between variables is important because it helps understand the relationship between them. Correlation reveals whether and

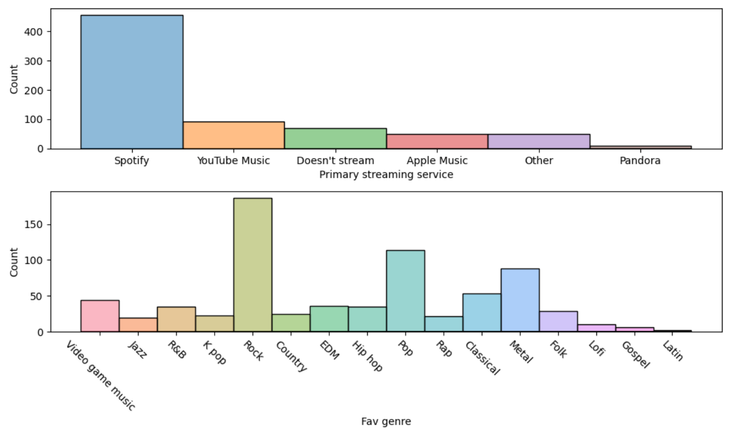


Figure 3.5: Genre and streaming platform preferences

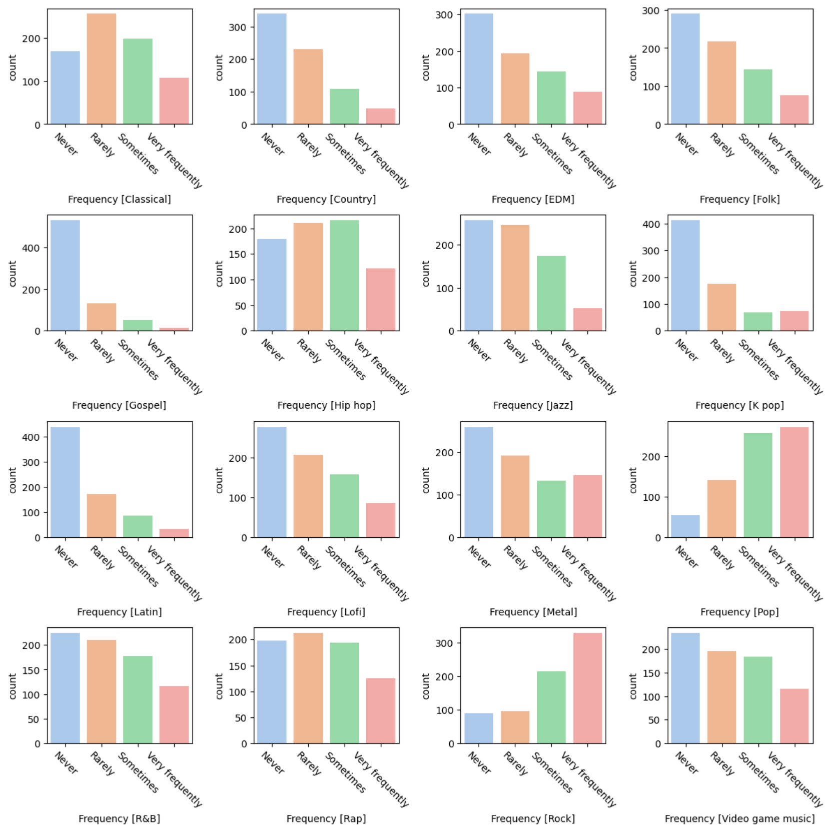


Figure 3.6: Listening frequency by genre

how strongly two variables are related, if trends exist in the data, and the possibility to predict highly correlated variables.

For our numerical variables we make use of the Pearson correlation matrix, a statistical tool used to measure the linear relationship between multiple variables. It returns a matrix of values ranging from -1 to +1, where:

- Values close to +1 represent a positive correlation
- Values close to -1 represent a negative correlation
- Values close to 0 represent no correlation between variables

As seen in the heatmap displaying the correlation matrix in Figure 3.7, we can see a certain level of correlation between our mental health variables ('Anxiety', 'Depression', 'Insomnia' and 'OCD') , suggesting that these co-occur or are related. The rest of our variables show generally minuscule to no relationships with the rest of the numerical variables.

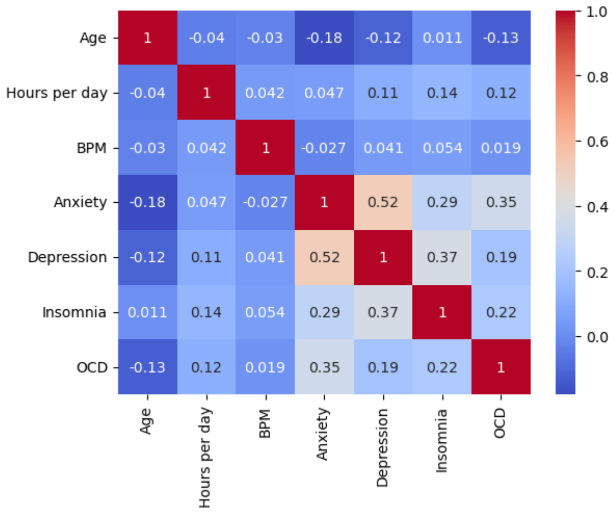


Figure 3.7: Correlation between numerical variables

As for our categorical variables, we use the chi-square tests of independence, which returns a p-value determining the relation between the variables. If the p-value is very small (usually below 0.05), we can infer a significant correlation between the variables.

We can observe in Figure 3.8 that our target variable has a strong relation with 'While working' and 'Instrumentalist', while also having a borderline association with 'Composer'. From this, we can determine a relation between music-involved testers and the music effect.

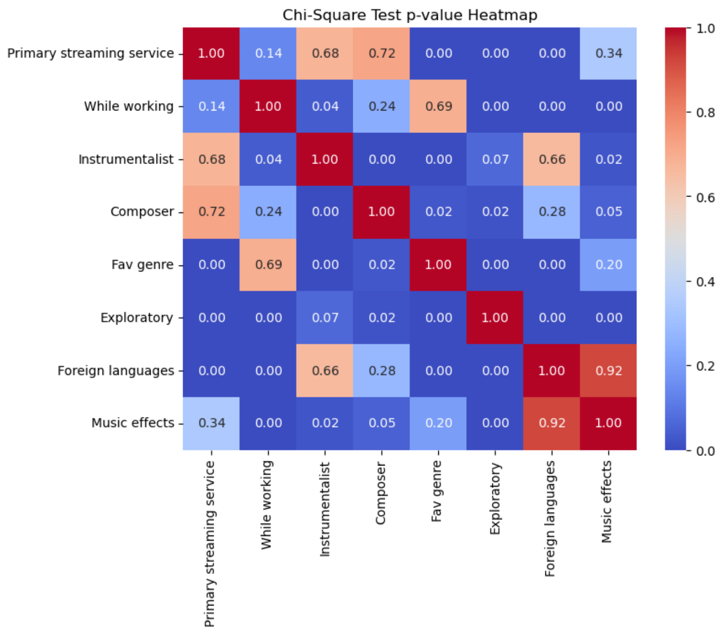


Figure 3.8: Correlation between categorical variables

Chapter 4

Data Preparation

This chapter details the crucial steps taken to clean, transform, and prepare the raw data for machine learning model training.

4.1 Feature Selection

As mentioned earlier in the report, we will be dropping ‘Timestamp’ (not relevant to our model) and ‘Permissions’ (it’s a constant throughout the entire dataset).

4.2 Missing values

The presence of missing values is a common challenge in real-world datasets, and most machine learning algorithms cannot directly process them. Therefore, addressing missing data is a mandatory step in data preparation.

The rows where ‘Music effects’ is missing will be removed, as they pose no significance to the intended model. As for the numerical variables, we’ll be applying scikit-learn’s SimpleImputer to fill the missing values with the mean of the column.

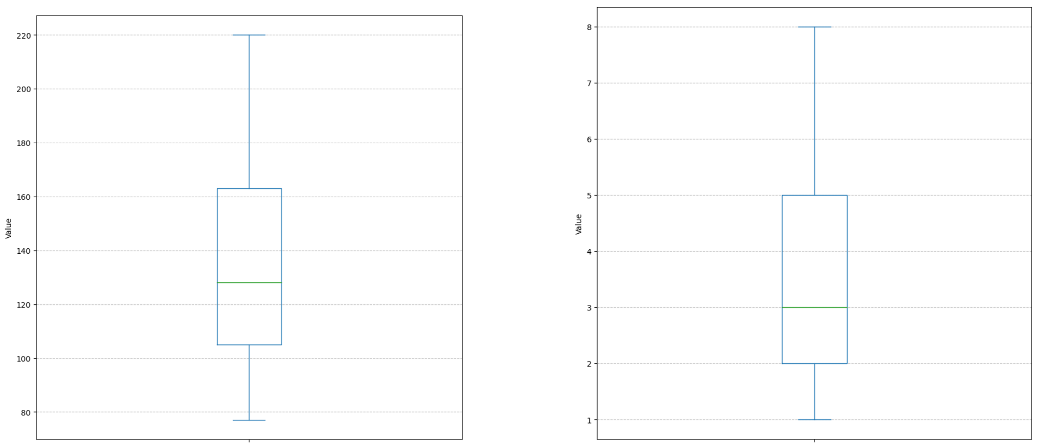
```
from scipy.stats.mstats import winsorize

df = data2
df["BPM"] = winsorize(data2["BPM"].values, limits
    =(0.05, 0.15))
df["Hours␣per␣day"] = winsorize(data2["Hours␣per␣day"].
    values, limits=(0.05, 0.055))
```

```
for col in categorical_cols:
    if df[col].isnull().any():
        mode_val = df[col].mode()[0]
        df[col].fillna(mode_val, inplace=True)
```

4.3 Outliers

For the outliers identified, we will be using Winsorization to "remove" them. This method replaces extreme values with less extreme values at specified percentiles. After applying this method to ‘BPM’ and ‘Hours per day’ we get the a better distribution of the values as seen in Fig 4.1.



(a) ‘BPM’ (b) ‘Hours per day’

Figure 4.1: Variables without outliers after Winsorization

4.4 Standardization and Transformation

Standardization is the process of rescaling data so that it has a mean equal to 0, and a standard deviation equal to 1. We do standardization because many models assume or are sensitive to the scale of input features. For our numerical features we apply scikit-learn’s StandardScaler to achieve this.

For our categorical attributes we need to encode them. Encoding is the process of converting categorical variables into a numerical format that models can understand, making them usable. We will be using a mix for encoders for these variables:

- **One-Hot Encoding:** it creates a new binary column for each category. Good for nominal variables.
- **Ordinal Encoding:** it maps categories to integers in a respecting order. Good for truly ordered categories.

For outliers, standardization, and transformation a list of pipelines were created in order to apply all methods needed for our data:

```
from sklearn.preprocessing import LabelEncoder,
    OrdinalEncoder, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
import pandas as pd

binary_cols = ['While␣working', 'Instrumentalist', '
    Composer', 'Exploratory', 'Foreign␣languages']

nominal_cols = ['Primary␣streaming␣service', 'Fav␣genre
    ']

ordinal_cols = [
    'Frequency␣[Classical]',
    'Frequency␣[Country]',
    'Frequency␣[EDM]',
    'Frequency␣[Folk]',
    'Frequency␣[Gospel]',
    'Frequency␣[Hip␣hop]',
    'Frequency␣[Jazz]',
    'Frequency␣[K␣pop]',
    'Frequency␣[Latin]',
    'Frequency␣[Lofi]',
    'Frequency␣[Metal]',
    'Frequency␣[Pop]',
    'Frequency␣[R&B]',
    'Frequency␣[Rap]',
    'Frequency␣[Rock]',
    'Frequency␣[Video␣game␣music]'
]
```

```
frequency_order = ["Never", "Rarely", "Sometimes", "
    Very␣frequently"]

X = df.drop(columns="Music␣effects")
y = df['Music␣effects']

binary_transformer = Pipeline([
    ('ordinal', OrdinalEncoder(categories=[['No', 'Yes'
        ]] * len(binary_cols), handle_unknown='
        use_encoded_value', unknown_value=-1))
])

nominal_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore',
        drop='first'))
])

ordinal_transformer = Pipeline([
    ('ordinal', OrdinalEncoder(categories=[
        frequency_order] * len(ordinal_cols),
        handle_unknown='use_encoded_value',
        unknown_value=-1))
])

numerical_pipeline = Pipeline([
    ("imputer", SimpleImputer()),
    ("scaler", StandardScaler())
])

le_target = LabelEncoder()
y = le_target.fit_transform(y)
```

```
from sklearn.compose import ColumnTransformer

preprocessing_pipeline = ColumnTransformer([
    ("numerical", numerical_pipeline, numerical_cols),
    ('bin', binary_transformer, binary_cols),
    ('nom', nominal_transformer, nominal_cols),
    ('ord', ordinal_transformer, ordinal_cols)
])
```

Chapter 5

Theory

5.1 Classification Models

5.1.1 Logistic Regression

Logistic Regression is a statistical model used for binary or multiclass classification tasks. It operates as a generalized linear model that estimates the probability of an instance belonging to a particular class. For binary classification, it uses the sigmoid (or logistic) function to map any real-valued number into a probability between 0 and 1. For multiclass problems, Logistic Regression can extend its capability through strategies like One-vs-Rest (OvR) or Multinomial Logistic Regression. OvR involves training a separate binary classifier for each class against all other classes, while Multinomial Logistic Regression (also known as softmax regression) directly models the probabilities of each class, estimating them using a softmax function. The model learns a set of weights for each feature, which are then used to calculate a score for each class, with the highest score indicating the predicted class.

```
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA

logistic_pipeline = Pipeline([
    ("preprocessing", preprocessing_pipeline),
    ("pca", PCA(n_components=40)),
    ("logistic", LogisticRegression(max_iter=10000))
])
```

5.1.2 Support Vector Classifier

Suport Vector Classifier (SVC) is a powerful supervised learning model used for classification. It seeks to find the optimal hyperplane that best separates data points of different classes in a high-dimensional space. For linearly separable data, the hyperplane maximizes the margin (the distance between the hyperplane and the nearest data points from each class, known as support vectors). For non-linearly separable data, SVC uses the "kernel trick," which implicitly maps the input features into a higher-dimensional feature space where a linear separation might be possible. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid.

```
from sklearn.svm import SVC

svc_pipeline = Pipeline([
    ("preprocessing", preprocessing_pipeline),
    ("pca", PCA(n_components=40)),
    ("classifier", SVC(gamma='auto'))
])
```

5.1.3 K-Nearest Neighbors Classifier

K-Nearest Neighbors (KNN) Classifier is a non-parametric, instance-based learning algorithm used for classification. Its fundamental principle is to classify a new data point based on the majority class among its 'K' nearest neighbors in the feature space. The 'distance' between data points is typically measured using metrics such as Euclidean distance. When a new data point needs to be classified, the algorithm identifies the 'K' training examples closest to it. The class label most frequent among these 'K' neighbors is then assigned to the new data point.

```
from sklearn.neighbors import KNeighborsClassifier

knn_pipeline = Pipeline([
    ("preprocessing", preprocessing_pipeline),
    ("pca", PCA(n_components=40)),
    ("classifier", KNeighborsClassifier(n_neighbors=6))
])
```

5.1.4 Random Forest Classifier

Random Forest Classifier is an ensemble learning method that belongs to the bagging (Bootstrap Aggregating) family of algorithms. It operates by constructing a multitude of decision trees during training and outputs the mode of the classes (for classification) predicted by the individual trees. Each tree in the forest is built from a random subset of the training data and, at each split in the tree, a random subset of features is considered.

```
from sklearn.ensemble import RandomForestClassifier

rfc_pipeline = Pipeline([
    ("preprocessing", preprocessing_pipeline),
    ("pca", PCA(n_components=40)),
    ("classifier", RandomForestClassifier(n_estimators
        =300, max_depth=8, random_state=0))
])
```

Chapter 6

Results

This chapter presents the findings from the model training and evaluation, including performance metrics and insights derived from the models.

6.1 Logistic Regression

The Logistic Regression model had the following results with the test set:

	Predicted Improve	**Predicted No effect**	**Predicted Worsen**
Actual Improve	53	1	0
Actual No effect	15	2	0
Actual Worsen	1	1	0

Table 6.1: Confusion Matrix for Logistic Regression

Table 6.2: Classification Report for Logistic Regression

Class	**Precision**	**Recall**	**F1-Score**
Improve	0.77	0.98	0.86
No effect	0.50	0.12	0.19
Worsen	0.00	0.00	0.00
Accuracy		0.75	
Cohen's kappa		0.14	

6.2 Support Vector Classifier

The Support Vector Classifier had the following results with the test set:

	Predicted Improve	**Predicted No effect**	**Predicted Worsen**
Actual Improve	54	0	0
Actual No effect	17	0	0
Actual Worsen	2	0	0

Table 6.3: Confusion Matrix for Support Vector Classifier

Table 6.4: Classification Report for Support Vector Classifier

Class	**Precision**	**Recall**	**F1-Score**
Improve	0.74	1	0.85
No effect	0.00	0.00	0.00
Worsen	0.00	0.00	0.00
Accuracy		0.74	
Cohen's kappa		0.00	

6.3 K-Nearest Neighbors Classifier

The K-Nearest Neighbors had the following results with the test set:

	Predicted Improve	**Predicted No effect**	**Predicted Worsen**
Actual Improve	53	1	0
Actual No effect	17	0	0
Actual Worsen	2	0	0

Table 6.5: Confusion Matrix for K-Nearest Neighbors Classifier

6.4 Random Forest Classifier

The Random Forest Classifier had the following results with the test set:

Table 6.6: Classification Report for K-Nearest Neighbors Classifier

Class	Precision	Recall	F1-Score
Improve	0.74	0.98	0.84
No effect	0.00	0.00	0.00
Worsen	0.00	0.00	0.00
Accuracy		0.73	
Cohen's kappa		-0.02	

	Predicted Improve	Predicted No effect	Predicted Worsen
Actual Improve	53	1	0
Actual No effect	17	0	0
Actual Worsen	2	0	0

Table 6.7: Confusion Matrix for Random Forest Classifier

Table 6.8: Classification Report for Random Forest Classifier

Class	Precision	Recall	F1-Score
Improve	0.74	0.98	0.84
No effect	0.00	0.00	0.00
Worsen	0.00	0.00	0.00
Accuracy		0.73	
Cohen's kappa		-0.02	

6.5 Best Model

Having seen the results of all the models, Logistic Regression appears to be the "best" possible model out of the bunch. Having the best Cohen's Kappa value, and a better performance with the 'No effect' category. The results for it with the validation set are as follows:

6.6 Discussion of Results

While Logistic Regression is the "best" model in this set due to its slightly better performance on Class 1 and a positive Cohen's Kappa, all models demonstrate severe limitations, particularly with minority classes. We can

	Predicted Improve	**Predicted No effect**	**Predicted Worsen**
Actual Improve	53	2	0
Actual No effect	14	3	0
Actual Worsen	1	0	0

Table 6.9: Confusion Matrix for Logistic Regression with Validation Set

Table 6.10: Classification Report for Logistic Regression with Validation Set

Class	**Precision**	**Recall**	**F1-Score**
Improve	0.78	0.96	0.86
No effect	0.60	0.18	0.27
Worsen	0.00	0.00	0.00
Accuracy		0.77	
Cohen's kappa		0.17	

observe poor performance on 'No effect' and 'Worsen', to the point of no predicting the second one in any of the models. Cohen's Kappa values for most models suggest a poor performance, and even worse than chance with K-Nearest Neighbors and Random Forest Classifier.

Chapter 7

Conclusion

This project aimed to classify the impact of music on mental health based on self-reported survey results, predicting whether music listening habits 'Improve', have 'No effect', or 'Worsen' an individual's mental well-being. For this, we developed and evaluated several classification models.

The analysis revealed that while Logistic Regression emerged as the "best" model among those tested, primarily due to its slightly better performance on the 'No effect' category and a positive Cohen's Kappa value, all models demonstrated significant limitations. A consistent challenge across all models was their poor performance on the minority classes, particularly the 'Worsen' category, which was often not predicted at all. The generally low Cohen's Kappa values, and even negative values for K-Nearest Neighbors and Random Forest Classifiers, suggest that the models' performance was often poor or no better than random chance, especially when considering imbalanced class distributions.

The study acknowledges several limitations. These include potential biases due to the self-reported nature and quality of the survey data, which likely contributed to the observed class imbalance where positive or neutral experiences were reported more frequently.

Bibliography

[1] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview, 2020. URL https://arxiv.org/abs/2008.05756.

[2] Catherine Rasgaitis. Music & mental health survey results, 2025. URL https://www.kaggle.com/datasets/catherinerasgaitis/mxmh-survey-results.

[3] scikit-learn developers. sklearn.neighbors.kneighborsclassifier — scikit-learn documentation, 2024. URL https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[4] scikit-learn developers. sklearn.ensemble.randomforestclassifier — scikit-learn documentation, 2024. URL https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[5] scikit-learn developers. sklearn.svm.svc — scikit-learn documentation, 2024. URL https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[6] scikit-learn developers. Logisticregression, 2024. URL https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.