Student: Giang Bui
ID: 37306207
Date: 09/06/2023

DATA 420- ASSIGNMENT 2

PREFACE

For this task, our focus will be on examining a compilation of datasets known as the Million Song Dataset (MSD). This project, originally started by The Echo Nest and LabROSA, aimed to comprehend the audio and textual components of recorded music. The Echo Nest, which emerged as a research spin-off from the MIT Media Lab, was later acquired by Spotify for a substantial sum of 50 million Euro after a decade of operation.
The primary dataset comprises essential information such as the song ID, track ID, artist ID, and 51 additional attributes. These additional attributes encompass details like the year, title, artist tags, and various audio characteristics such as loudness, beat, tempo, and time signature. Our attention will be directed towards analyzing the Taste Profile and All Music datasets.
Each question's solution code is provided individually and is thoroughly commented for clarity. This report showcases the methodology employed to solve each problem and delves into the data and its intriguing aspects.

DATA PROCESSING

Question 1. Defining the structures of the data:

The entire dataset is located within the HDFS directory at hdfs:///data/msd/. This directory is organized into four main parts: 'audio', 'genre', 'main', and 'tasteprofile'.
Within the 'audio' directory, there are three subdirectories: 'attributes', 'features', and 'statistics'. These subdirectories contain files stored in both csv and csv.gz formats. The 'genre' directory includes three tsv files, while the 'main/summary' directory has two files stored as csv.gz. Lastly, the 'tasteprofile' directory stores mismatched datasets and the file 'triplets.tsv', which is partitioned into multiple tsv.gz files.

The size of each folder are recored as table 1 below. It is evident that the 'audio' folder occupies the largest portion, amounting to 95.3% of the overall size, which is equivalent to 12.3 gigabytes. The 'tasteprofile' directory comes in second place, accounting for 490.4 megabytes, followed by 'main' and 'genre' with sizes of 174.4 megabytes and 30.1 megabytes respectively. The total size of the 'msd' folder is 12.9 gigabytes.

File	Size
Audio	12.3 G
Genre	30.1 M
Main	174.4 M
Tasteprofile	490.4 M
Total	12.9 G

Table 1. Size of MSD's Folders

The directory tree and the number of rows for each files are illustrate as table 2 below. The 'Audio' directory has the highest number of rows, primarily due to its inclusion of numerous attributes and features, surpassing the count of unique songs.

Table 2.Directory tree and the number of rows for each msd's file

Directory/file	Total rows count
|-- audio	49,153,611
|----- attributes	3,929
|----------*msd-jmir-area-of-moments-all-v1.0.attributes.csv*	21
|----------*msd-jmir-lpc-all-v1.0.attributes.csv*	21
|----------*msd-jmir-methods-of-moments-all-v1.0.attributes.csv*	11
|----------*msd-jmir-mfcc-all-v1.0.attributes.csv*	27
|----------*msd-jmir-spectral-all-all-v1.0.attributes.csv*	17
|----------*msd-jmir-spectral-derivatives-all-all-v1.0.attributes.csv*	17
|----------*msd-marsyas-timbral-v1.0.attributes.csv*	125
|----------*msd-mvd-v1.0.attributes.csv*	421
|----------*msd-rh-v1.0.attributes.csv*	61
|----------*msd-rp-v1.0.attributes.csv*	1,441
|----------*msd-ssd-v1.0.attributes.csv*	169
|----------*msd-trh-v1.0.attributes.csv*	421
|----------*msd-tssd-v1.0.attributes.csv*	1,177
|----- features	48,966,420
|----------msd-jmir-area-of-moments-all-v1.0.csv	266,223
|----------part-00000.csv.gz	
|----------...	
|----------part-00007.csv.gz	
|----------msd-jmir-lpc-all-v1.0.csv	203,534
|----------part-00000.csv.gz	
|----------...	
|----------part-00007.csv.gz	
|----------msd-jmir-methods-of-moments-all-v1.0.csv	156,722
|----------part-00000.csv.gz	
|----------...	
|----------part-00007.csv.gz	
|----------msd-jmir-mfcc-all-v1.0.csv	258,829
|----------part-00000.csv.gz	
|----------...	
|----------part-00007.csv.gz	
|----------msd-jmir-spectral-all-all-v1.0.csv	200,436
|----------part-00000.csv.gz	
|----------part-00001.csv.gz	
|----------...	
|----------part-00007.csv.gz	
|----------msd-jmir-spectral-derivatives-all-all-v1.0.csv	200,436
|----------part-00000.csv.gz	
|----------...	
|----------part-00007.csv.gz	
|----------msd-marsyas-timbral-v1.0.csv	1,685,162
|----------part-00000.csv.gz	
|----------...	
|----------part-00007.csv.gz	
|----------msd-mvd-v1.0.csv	5,280,662
|----------part-00000.csv.gz	
|----------...	
|----------part-00007.csv.gz	
|----------msd-rh-v1.0.csv	899,531
|----------part-00000.csv.gz	
|----------...	
|----------part-00007.csv.gz	
|----------msd-rp-v1.0.csv	17,359,923
|----------part-00000.csv.gz	
|----------...	
|----------part-00007.csv.gz	
|----------msd-ssd-v1.0.csv	2,270,027
|----------part-00000.csv.gz	
|----------...	
|----------part-00007.csv.gz	
|----------msd-trh-v1.0.csv	5,373,985
|----------part-00000.csv.gz	
|----------...	
|----------part-00007.csv.gz	
|----------msd-tssd-v1.0.csv	14,810,950
|----------part-00000.csv.gz	

|----------...|
|----------part-00007.csv.gz|
|----- statistics	
|----------sample_properties.csv.gz	183,262
|-- genre	1,103,077
|----- msd-MAGD-genreAssignment.tsv	422,714
|----- msd-MASD-styleAssignment.tsv	273,936
|----- msd-topMAGD-genreAssignment.tsv	406,427
|-- main	
|----- summary	
|----------analysis.csv.gz	239,762
|----------metadata.csv.gz	480,546
|-- tasteprofile	1,694,752
|----- mismatches	20,032
|----------sid_matches_manually_accepted.txt	938
|----------sid_mismatches.txt	19,094
|----- *triplets.tsv*	1,674,720
|----------part-00000.csv.gz	210,041
|----------part-00001.csv.gz	209,199
|----------part-00002.csv.gz	208,896
|----------part-00003.csv.gz	209,050
|----------part-00004.csv.gz	209,315
|----------part-00005.csv.gz	210,353
|----------part-00006.csv.gz	209,326
|----------part-00007.csv.gz	208,540

The utilization of the repartition method can be advantageous as it expands the number of partitions, resulting in an augmented level of parallelism and faster completion of the job. Nonetheless, it is essential to carefully assess the trade-off between the performance benefits gained through parallelization and the associated shuffle cost. Repartitioning demands additional resources due to this shuffle cost, and a thorough consideration of these factors is necessary.

Question 2. Data preprocessing

a. Taste Profile dataset:

In this section, our focus is on exploring the 'Taste Profile' dataset, which consists of actual user-play counts obtained from undisclosed partners. This dataset includes songs that have already been matched with the Million Song Dataset (MSD). Within the 'tasteprofile' folder, there is a subfolder called 'mismatches' that serves as a log, identifying instances where tracks were incorrectly associated with the wrong songs or where certain matches were not identified at all. It is necessary to filter out the mismatched songs from further analysis.

To begin, we convert the *'sid_matches_manually_accepted.txt'* and *'sid_mismatches.txt'* files into dataframes, organizing them into rows and columns. Next, we execute a left anti-join operation between these two dataframes to filter out the mismatches that were not accepted. Subsequently, we perform another left anti-join between the triplets data and the results from the previous step, obtaining the good triplets without any mismatches. After the cleaning step, the final count of tracks is 45,795,111, which represents 94.67% of the total triplets. This means that 5.33% of the triplets data were removed due to mismatched issues.

The number of tracks from each data frame are shown as table 3 below.

Dataframe	Number of Tracks
matches_manually_accepted	488
mismatches	19,094
triplets	48,373,586
triplets_not_mismatched	45,795,111

Table 3. The number of tracks from each 'Taste Profile' dataset.

b. Automated dataframe creation for files under audio/features directory

First, we create schema table for each dataset under audio/features directory base on the metadata attribute csv files in folder /audio/attributes/. For example, the schema table of '*msd-jmir-area-of-moments-all-v1.0*' is shown as table 4 below.

name	type
Area_Method_of_Moments_Overall_Standard_Deviation_1	real
Area_Method_of_Moments_Overall_Standard_Deviation_2	real
Area_Method_of_Moments_Overall_Standard_Deviation_3	real
Area_Method_of_Moments_Overall_Standard_Deviation_4	real
Area_Method_of_Moments_Overall_Standard_Deviation_5	real
Area_Method_of_Moments_Overall_Standard_Deviation_6	real
Area_Method_of_Moments_Overall_Standard_Deviation_7	real
Area_Method_of_Moments_Overall_Standard_Deviation_8	real
Area_Method_of_Moments_Overall_Standard_Deviation_9	real
Area_Method_of_Moments_Overall_Standard_Deviation_10	real
Area_Method_of_Moments_Overall_Average_1	real
Area_Method_of_Moments_Overall_Average_2	real
Area_Method_of_Moments_Overall_Average_3	real
Area_Method_of_Moments_Overall_Average_4	real
Area_Method_of_Moments_Overall_Average_5	real
Area_Method_of_Moments_Overall_Average_6	real
Area_Method_of_Moments_Overall_Average_7	real
Area_Method_of_Moments_Overall_Average_8	real
Area_Method_of_Moments_Overall_Average_9	real
Area_Method_of_Moments_Overall_Average_10	real
MSD_TRACKID	string

Table 4. Schema of 'msd-jmir-area-of-moments-all-v1.0'

Base on the created schema table, we generate its schema by mapping the 'name' and 'type' from its schema table. Table 5 illustrates the schema of file '*msd-jmir-area-of-moments-all-v1.0*' and the simple version which the name of each column was shorten.

```
actual_schema = StructType([
  StructField('Area_Method_of_Moments_Overall_Standard_Deviation_1', DoubleType(), True),
  StructField('Area_Method_of_Moments_Overall_Standard_Deviation_2', DoubleType(), True),
  …
  StructField('Area_Method_of_Moments_Overall_Standard_Deviation_10', DoubleType(), True),
  StructField('Area_Method_of_Moments_Overall_Average_1', DoubleType(), True),
  StructField('Area_Method_of_Moments_Overall_Average_2', DoubleType(), True),
  …
  StructField('Area_Method_of_Moments_Overall_Average_10', DoubleType(), True),
  StructField('MSD_TRACKID', StringType(), True)
])
```

```
simple_schema = StructType([
  StructField('F000', DoubleType(), True),
  StructField('F001', DoubleType(), True),
  StructField('F002', DoubleType(), True),
  …
  StructField('F019', DoubleType(), True),
  StructField('ID', StringType(), True)
])
```

Table 5. Schema of file 'msd-jmir-area-of-moments-all-v1.0' and its simple version.

AUDIO SIMILARITY

Question 1. Feature data and genre data exploration

a. Feature data

Each feature dataset has a different levels of details. In this section we will look into a single feature file from the features directory and perform descriptive analysis. First, a dataset which has 994,623 rows named *'msd-jmir-mfcc-all- v1.0'* from the audio/features directory is randomly chose and its schema is defined and the respective dataset is imported from its csv file. The dataframe has 27 columns which the first 26 columns are numeric, and the last column 'MSD_TRACKID' serves as identifier. The actual column names and it's shorten versions is shown as table 6 below.

Actual name	Short name
MFCC_Overall_Standard_Deviation_1	F000
MFCC_Overall_Standard_Deviation_2	F001
MFCC_Overall_Standard_Deviation_3	F002
MFCC_Overall_Standard_Deviation_4	F003
MFCC_Overall_Standard_Deviation_5	F004
MFCC_Overall_Standard_Deviation_6	F005
MFCC_Overall_Standard_Deviation_7	F006
MFCC_Overall_Standard_Deviation_8	F007
MFCC_Overall_Standard_Deviation_9	F008
MFCC_Overall_Standard_Deviation_10	F009
MFCC_Overall_Standard_Deviation_11	F010
MFCC_Overall_Standard_Deviation_12	F011
MFCC_Overall_Standard_Deviation_13	F012
MFCC_Overall_Average_1	F013
MFCC_Overall_Average_2	F014
MFCC_Overall_Average_3	F015
MFCC_Overall_Average_4	F016
MFCC_Overall_Average_5	F017
MFCC_Overall_Average_6	F018
MFCC_Overall_Average_7	F019
MFCC_Overall_Average_8	F020
MFCC_Overall_Average_9	F021
MFCC_Overall_Average_10	F022
MFCC_Overall_Average_11	F023
MFCC_Overall_Average_12	F024
MFCC_Overall_Average_13	F025
MSD_TRACKID	ID',

Table 6. Actual column names and it's shorten versions of 'msd-jmir-mfcc-all- v1.0' file.

The descriptive statistics for each feature column 'msd-jmir-mfcc-all- v1.0' is shown as table 7.

summary	F000	F001	F002	F003	F004	F005	F006	F007	F008	F009
count	994,623	994,623	994,623	994,623	994,623	994,623	994,623	994,623	994,623	994,623
mean	46.24	5.61	4.30	3.23	2.69	2.44	2.24	1.94	1.87	1.83
stddev	23.15	1.89	1.21	0.73	0.59	0.50	0.42	0.35	0.33	0.32
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
max	544.00	51.19	24.48	20.28	21.02	33.46	19.15	23.39	23.08	26.12
summary	F010	F011	F012	F013	F014	F015	F016	F017	F018	F019
count	994,623	994,623	994,623	994,623	994,623	994,623	994,623	994,623	994,623	994,623
mean	1.70	1.58	1.44	-115.59	11.39	-0.55	4.07	-0.30	0.97	-0.78
stddev	0.32	0.28	0.25	43.37	6.76	4.04	2.43	1.75	1.54	1.21
min	0.00	0.00	0.00	-1,150.00	-115.70	-37.68	-33.08	-19.74	-16.39	-14.68
max	37.27	25.74	19.38	-50.81	62.30	58.51	36.08	49.66	78.78	42.52
summary	F020	F021	F022	F023	F024	F025				
count	994,623	994,623	994,623	994,623	994,623	994,623				
mean	1.07	-0.62	0.99	-0.67	0.29	-0.55				
stddev	1.07	0.96	0.80	0.75	0.63	0.57				
min	-15.24	-38.85	-7.52	-11.13	-7.13	-24.18				
max	55.60	53.53	54.79	85.05	59.06	28.42				

Table 7. Descriptive statistics for each feature column 'msd-jmir-mfcc-all- v1.0' dataset.

The correlations between 'msd-jmir-mfcc-all- v1.0' features are examined and illustrated as figure 1 and table 8. As we can see, the pairs (F010, F01), (F011, F012), (F009, F008) are the top three with highest correlations. Based on the observed correlation issue, it is recommended to exclude the features **F011** and **F008** from the dataset before training the model. Removing these two features can help mitigate the potential impacts of their correlation on the model's performance.

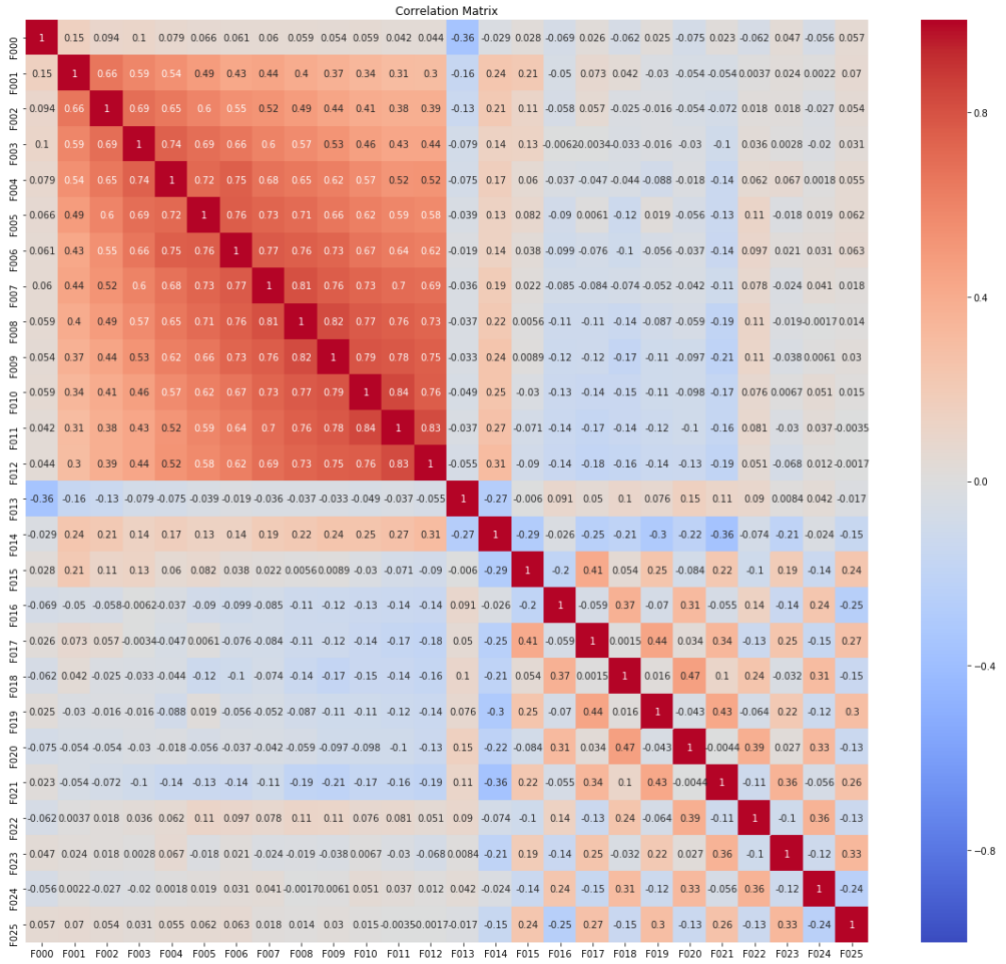


Figure 1. The correlations between 'msd-jmir-mfcc-all- v1.0' features.

Feature	Feature	Correlation score
F010	**F011**	0.8375
F011	F012	0.8269
F009	**F008**	0.8220
F008	F007	0.8094
F010	F009	0.7913
F011	F009	0.7755
F007	F006	0.7731
F008	F010	0.7729
F012	F010	0.7643
F006	F005	0.7617

Table 8. Top 10 pairs which are most correlated in 'msd-jmir-mfcc-all- v1.0' dataset.

b. Genre data

There are 3 tsv files under 'genre' directory. We will examine the *'msd-MAGD-genreAssignment.tsv'* file by import it into genre_data dataframe. The schema for genre including 2 columns 'track_id' and 'genre'. A glance at the *genre_data* is shown as table 9.

track_id	genre
TRAAAAK128F9318786	Pop_Rock
TRAAAAV128F421A322	Pop_Rock
TRAAAAW128F429D538	Rap
TRAAABD128F429CF47	Pop_Rock
TRAAACV128F423E09E	Pop_Rock

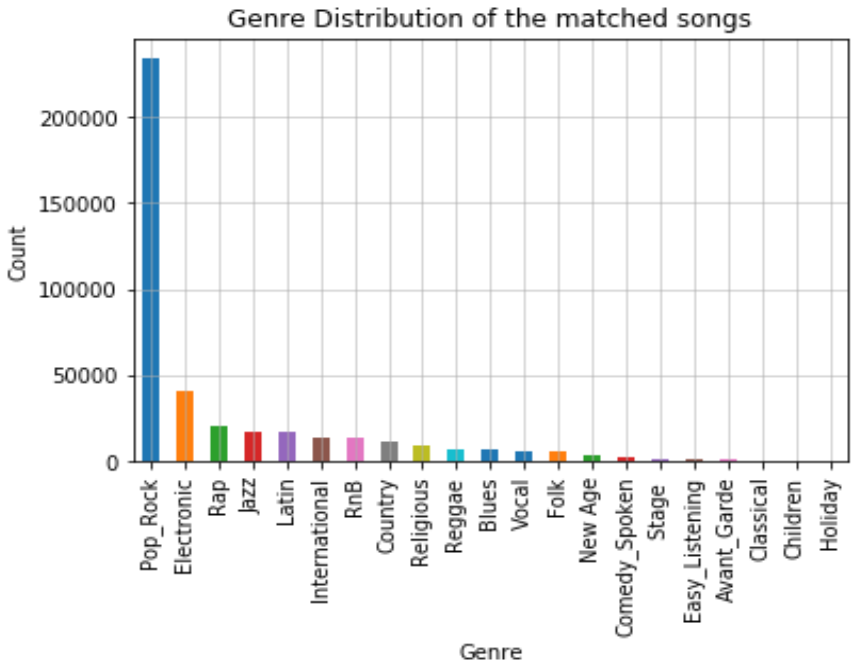
Table 9. Genre_data's first 5 rows.

Performing groupBy ‘genre’ and aggregate ‘count’, we observe the total number of tracks for each genre, which are recorded as the table 10 and its distribution is visualised as figure 2.

Table 10. Genre's total number of tracks.

genre	count
Pop_Rock	234,107
Electronic	40,430
Rap	20,606
Jazz	17,673
Latin	17,475
International	14,094
RnB	13,874
Country	11,492
Religious	8,754
Reggae	6,885
Blues	6,776
Vocal	6,076
Folk	5,777
New Age	3,935
Comedy_Spoken	2,051
Stage	1,604
Easy_Listening	1,533
Avant_Garde	1,000
Classical	542
Children	468
Holiday	198

Figure 2. Genre's track distribution



It is evident that the 'Pop_Rock' genre is the most popular, encompassing a total of 234,107 distinct tracks. Following behind are the genres 'Electronic' and 'Rap' with only 40,430 and 20,606 tracks respectively. On the other end of the spectrum, 'Classical', 'Children', and 'Holiday' are the least popular genres, containing only 542, 468, and 198 tracks respectively.

We perform left join the feature data with the genre data on feature ID = genre ID so that all the tracks are mapped with its respective genre.

Question 2. Binary classification

For the binary classification task, we will construct three models: Logistic Regression, Random Forest, and Support Vector Machine. It is important to note that there exists a trade-off between accuracy and interpretability, as depicted in Figure 3.

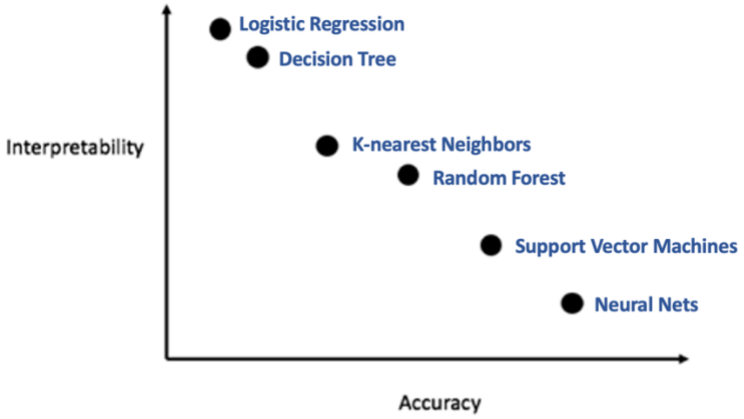


Figure 3. trade-off between model’s accuracy and interpretability.

Below are the descriptions and reasons for choosing the three models for classification task:

Method	Description	Reason chosen
Logistic Regression	The method model is used for binary classification tasks, where the goal is to predict the probability of an instance belonging to a specific class using logistic or sigmoid function. It maps any real-valued input to a value between 0 and 1. Assumption: First, the method assumes a linear relationship between the outcome and the log-odds of the predictors, which allows the estimation of impact of each predictors on the predicted probability. Second, it assumes that the observations are independent, there is no multicollinearity among predictors, and the log-odds relationship is correctly specified.	This method provides a straightforward approach. Although the model accuracy is not highly expected, the model is simple, highly interpretable. To be more specific, the method use elastic net regularization to scale up or down the coefficients of the predictors. It also do feature selection, which is great when there are many predictors in the dataset. We use this method as a baseline for other more complex model performance. The method can handle imbalanced datasets effectively.
Random Forest	The method uses ensemble learning algorithm, where multiple decision trees are trained on different subset of the training data. This randomness helps introduce diversity among the tree. The method uses bootstrap aggregating technique to create subsets of data for training each decision tree. For classification problem, it uses majority voting to determine the final predicted class. The method can effectively capture the nonlinearities between predictors and outcome. It can handle a wide variety of data types and robust to outliers.	The model performance can be improved by tuning the hyper parameters. The model estimate the variable importance which is helpful for feature selection and we can estimate the impact of each variable to the prediction. Random Forest can be easily parallelized, allowing for efficient training on large datasets by utilizing multiple processors or cores. The model can cope with non-linear relationships and high dimensional data.
Support Vector Machine	This method uses a powerful machine learning algorithm, which find a decision boundary that maximise the distance between decision boundary and the closest data points of each class in the feature space. SVM is capable of handling linear or nonlinear data by using kernel trick technique, which allows SVM to implicitly transform the data into a higher-dimensional feature space, where it becomes easier to find a linear decision boundary. SVM identifies support vectors which are subset of the training data points, which are the closest to the decision boundary. These vectors can influence the performance of the model. SVM incorporates a regularization parameter (C) that controls the trade-off between classification error and maximizing the margin.	SVM is chosen for many reasons. First, it is robust to overfitting because it can handle high-dimentional feature spaces effectively. Second, SVM provides interpretable results by indentifying the support vectors, which are important in model accuracy. Third, the method provides robust generalization and can capture both linear and nonlinear relationships.

Each model candidate has its strengths and considerations, and the selection depends on factors such as the nature of the data, desired interpretability, and the trade-off between accuracy and complexity.

To maximise the model performance, a series of pre-processing are employed to the audio features before training each model including duplication, missing data checks, track_ID remove, highly

correlated variables remove, binary coding for genre column. After train/test split, down sampling is employed for training set to address class imbalance issue.
There are 420,620 observations in the dataset. The data for classification is unbalanced, which the total observations of class 1 is 40,666, account for only 9.6681% overall. Meanwhile, class 0 comprises 379,954 observations, which accounts for 90.33% total dataset.
The total observations and the ratio for each class for training pre down sampling, after down sampling data, and test data are shown as table 11.
The purpose of down sampling method is to mitigating the bias towards the class 0 and ensure the model predict class 1 accurately. Also, down sampling prevents overfitting and improve the model's generalization performance. We only use down sampling for the training dataset.

	Train data (no sampling) Total observations: 336495		Train data (downsampling) Total observations: 97,670		Test data Total observations: 84,125	
Class	count	ratio	count	ratio	count	ratio
0	303,963	0.903321	65,138	0.666919	75,991	0.903311
1	32,532	0.096679	32,532	0.333081	8,134	0.096689

Table 11. The total observations and the ratio for each class.

Three methods: Logistic regression, random forest and support vector machine (SVM) are employed to train the model. The model performance metrics are evaluated and are shown as table 12.

Method	Metric	Logistic regression	Random Forest	Support Vector Machine
No-sampling	Actual count	84,125	84,125	84,125
	True Positive	1,217	205	0
	True Negative	75,291	75,911	75,991
	False Positive	700	80	0
	False Negative	6,917	7,929	8,134
	precision	0.6348	0.7193	NA
	recall	0.1496	0.0252	0.0000
	accuracy	0.9095	0.9048	0.9033
	auroc	0.7996	0.7510	0.7575
	F1_Score	0.2422	0.0487	NA
Down-sampling	Actual count	84,125	84,125	84,125
	True Positive	4,079	3,023	3,751
	True Negative	69,106	72,126	70,444
	False Positive	6,885	3,865	5,547
	False Negative	4,055	5,111	4,383
	precision	0.3720	0.4389	0.4034
	recall	0.5015	0.3716	0.4612
	accuracy	0.8700	0.8933	0.8820
	auroc	0.8009	0.7977	0.7897
	F1_Score	0.4272	0.4025	0.4304

Table 12. Model performance metrics.

Justify 'Accuracy' metric

As we can see that the accuracy is a bad metric in imbalanced class prediction. For example, the accuracy of Logistic regression model of no-sampling method is 91.07% but only 15.65% of total tracks are correctly classified (recall metric). Random Forrest model witnessed the similar issue. Meanwhile, SVM running errors when handling imbalanced class issue. F1_Score, Auroc, recall and precision are the more appropriate metrics to evaluate the models.

No-sampling versus down sampling methods

When changing from exact stratified sampling to downsampling method, the recall increased from 14.96% to 50.15% with the Logistic regression model, from 2.52% to 37.16% with the random forest which is a huge improvement. The recall of SVM for downsampling method is 4 model performance metrics 46.12%. It suggests that down sampling is a great preprocessing method for solving class imbalance problem.

Comparision between logistic regression, random forest and SVM models

To estimate the comprehensive evaluation of model's overall performance, we examine AUROC and F1 score, which strikes a balance between correctly identifying positive instances and minimizing false instances. Observed from the table 12, SVM has highest F1 score with 0.4304, followed closely behind is Logistic Regression with 0.4272 and random forest with 0.4025.

In terms of interpretability and computational time, logistic regression is better than the other two models.

In terms of AUROC metric, Logistic regression outperforms Random Forest and SVM with 0.8009 compared to 0.7977 and 0.7897 respectively.

Question 3. Hyperparameters tunning of your best performing binary classification model.

Based on the analysis of the three models from question 2, it can be concluded that Logistic Regression is considered the best model. This conclusion is based on the following factors: simplicity, high interpretability, lower computational cost, and comparable model performance to Random Forest and Support Vector Machine models.

We continue to tune the best model "Logistic Regression" with patamethers grid using 5-fold cross validation which presents in the table 13, and the model performance metrics are recorded in the table 14.

Parameter	Tuning value
elasticNetParam	0.0, 0.1,0.3, 0.5, 0.8, 1.0
maxIter	10, 100
regParam	0, 0.01, 0.5, 1.0

Table 13. Parameter grid for tuning Logistic Regression model.

Metric	Value
Actual count	84,125
True Positive	4,079
True Negative	69,106
False Positive	6,885
False Negative	4,055
precision	0.3720
recall	0.5015
accuracy	0.8700
auroc	0.8009
F1_Score	0.4272

Table 14. Model performance metrics after tunning Logistic Regression.

It suggests that there is almost no difference between the logistic model before and after tunning using 5-fold cross validation

Question 4. Multiclass classification: predict across all genres

To deal with multiclass classification, we consider the two strategies one-vs-one and one-vs-all.

One-vs-one: For this strategy, a binary classifier is trained for each pair of classes. In total, we have N*(N-1)/2 classifiers are trained. During prediction, each classifier predicts between two classes, and the class with most votes across all classifiers is chosen as the final prediction. The advantage of using this strategy is that it can handle effectively unbalanced class distributions, and not sensitive with the presence of noisy data. However, the drawback is that it is computational expensive.

One-vs-all: For this strategy, a binary classifier is trained for each class to distinguish it from the rest classes combined. For a problem with N classes, only N binary classifiers are trained, which is much less than the one-vs-one. During prediction, each classifier predicts either the positive class (the class it was trained for) or the negative class (all other classes combined). The class with the highest probability or confidence score is selected as the final prediction. The upside of this strategy is that it is efficient when the computational source is limited. Also, it can handle class imbalance and easier to parallelise.
One-vs-one is expected to perform better in multiclass classification problem. However, as the dataset size and number of classes are large, one-vs-all is chosen for solving this problem.

First, we convert the genre value into integer. The total observations of each class are shown as the table 15 below.

label	genre	count
0	Pop_Rock	237,649
1	Electronic	40,666
2	Rap	20,899
3	Jazz	17,775
4	Latin	17,504
5	RnB	14,314
6	International	14,194
7	Country	11,691
8	Religious	8,780
9	Reggae	6,931
10	Blues	6,801
11	Vocal	6,182
12	Folk	5,789
13	New Age	4,000
14	Comedy_Spoken	2,067
15	Stage	1,613
16	Easy_Listening	1,535
17	Avant_Garde	1,012
18	Classical	555
19	Children	463
20	Holiday	200

Table 15. Genre encoding and its number of observations

The dataset was subsequently divided into training and test sets using an 80/20 ratio. The training set underwent a down sampling processfor two classes 0 and 1 with ratio 0.1 and 0.5 respectively, and up sampling for the classes whose number of observations is smaller than 2000 (class 14 to 20). This preprocessing step will help tackle the class imbalance issue due to the domination of class 0 and 1 and subjugation of minor classes. The result is in shown in the table 16.

	Total		Train		Resampling Train		Test	
label	count	ratio	count	ratio	count	ratio	count	ratio
0	237,649	0.564997	190,119	0.565018	19,082	0.121457	47,530	0.564912
1	40,666	0.096681	32,532	0.096682	16,245	0.103400	8,134	0.096676
2	20,899	0.049686	16,719	0.049688	16,719	0.106417	4,180	0.049681
3	17,775	0.042259	14,219	0.042258	14,219	0.090504	3,556	0.042264
4	17,504	0.041615	14,003	0.041616	14,003	0.089129	3,501	0.041611
5	14,314	0.034031	11,451	0.034031	11,451	0.072886	2,863	0.034028
6	14,194	0.033745	11,355	0.033746	11,355	0.072275	2,839	0.033743
7	11,691	0.027795	9,352	0.027793	9,352	0.059526	2,339	0.027800
8	8,780	0.020874	7,023	0.020872	7,023	0.044701	1,757	0.020883
9	6,931	0.016478	5,544	0.016476	5,544	0.035288	1,387	0.016485
10	6,801	0.016169	5,440	0.016167	5,440	0.034626	1,361	0.016176
11	6,182	0.014697	4,945	0.014696	4,945	0.031475	1,237	0.014702
12	5,789	0.013763	4,631	0.013763	4,631	0.029476	1,158	0.013763
13	4,000	0.00951	3,199	0.009507	3,199	0.020362	801	0.009520
14	2,067	0.004914	1,653	0.004913	2,019	0.012851	414	0.004921
15	1,613	0.003835	1,290	0.003834	1,970	0.012539	323	0.003839
16	1,535	0.003649	1,227	0.003647	1,977	0.012584	308	0.003661
17	1,012	0.002406	809	0.002404	1,924	0.012246	203	0.002413
18	555	0.001319	443	0.001317	1,982	0.012615	112	0.001331
19	463	0.001101	370	0.001100	1,930	0.012284	93	0.001105
20	200	0.000475	159	0.000473	1,968	0.012526	41	0.000487

Table 16. Class size for training data (before and after resampling) and test data.

The model evaluation metrics are computed and shown in table 17 below. The model's performance can be considered moderate. The weighted precision, weighted recall, and accuracy scores show consistency in predictions across different classes. The F1 score, which considers both precision and recall, provides a comprehensive evaluation.

Metric	Value
weighted Precision	0.5448
weighted Recall	0.4837
accuracy	0.4837
weighted F1	0.4949

Table 17. Model performance metric of multiclass classification model

Figure 4 illustrates the Precision and Recall values for each class. Class 0 demonstrates the highest classification performance among the 21 classes. The use of up and down sampling techniques in the training dataset has contributed to an improvement in the overall weighted metrics. The model shows the ability to classify the minor classes such as 18, 19, and 20. However, challenges arise in classifying certain classes like 9 and 17, where both precision and recall values are 0, indicating difficulties in accurately predicting these classes.

Figure 4. PRECISION AND RECALL FOR 21 CLASSES

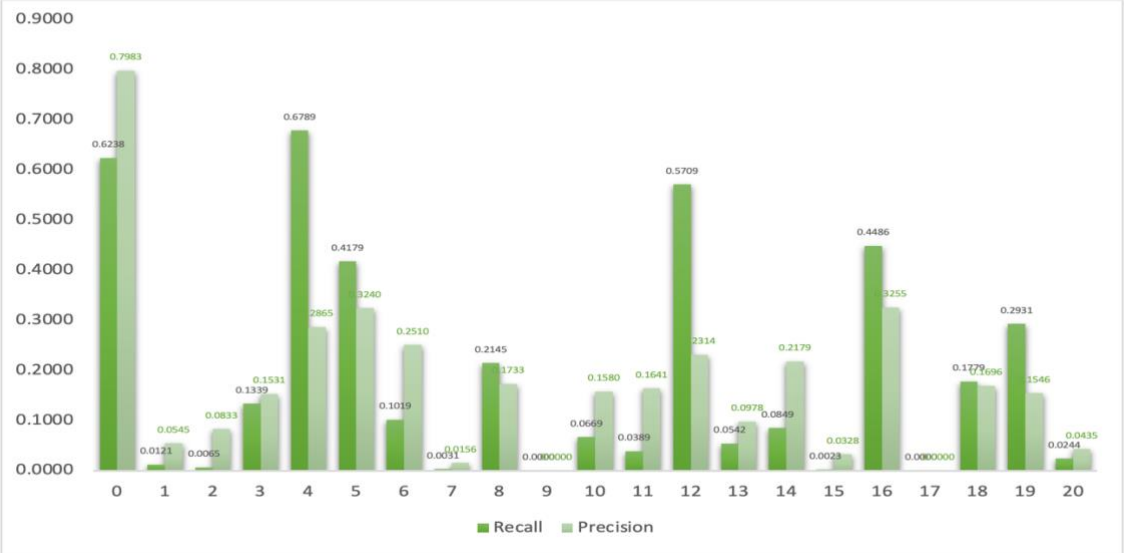


Figure 4. Precision and Recall for each class

SONG RECOMMENDATIONS

Question1. Exploring Taste Profile dataset

There are 378,310 unique songs and 1,019,318 unique users in the Taste Profile dataset. Top five most active users with their total number of unique songs and total plays is shown as the table 18. The user with ID "093cb74eb3c517c5179ae24caf0ebec51b24d2a2" is the most active user with 195 unique song played, which account for 0.0515% total number of songs, which is 378,310 songs.

user_id	song_count	play_count
093cb74eb3c517c5179ae24caf0ebec51b24d2a2	195	13,074
119b7c88d58d0c6eb051365c103da5caf817bea6	1362	9,104
3fa44653315697f42410a30cb766a4eb102080bb	146	8,025
a2679496cd0af9779a92a13ff7c6af5c81ea8c7b	518	6,506
d7d2d888ae04d16e994d6964214a1de81392ee04	1257	6,190

Table 18. Top five most active users in Taste Profile Dataset

Similarity, we have top five song with total number of users and total number of plays shown in table 19 below. The most played song is 'SOBONKR12A58A7A7E0' with 726,885 plays from 84,000 different users.

song_id	user_count	play_count
SOBONKR12A58A7A7E0	84,000	726,885
SOSXLTC12AF72A7F54	80,656	527,893
SOEGIYH12A6D4FC0E3	69,487	389,880
SOAXGDH12A8C13F8A1	90,444	356,533
SONYKOW12AB01849C9	78,353	292,642

Table 19. Top five most popular songs in Taste Profile Dataset

The song popularity is defined as the total number of plays for each song ID. The user active level is defined as the total number of plays for each user ID.

The descriptive statistics of song popularity and user active level are shown as table 20 below.

	count	mean	mode	min	25%	median	75%	max	stddev
user activity	1,019,318	128.82	13,074	3	30	64	152	13,074	175.44
Song poppularity	378,310	347.10	726,885	1	7	28	116	726,885	2978.61

Table 20. Descriptive statistics of song popularity and user active level.

Visualize the distribution of song popularity and the distribution of user activity are shown as figure 5 and figure 6 respectively. It suggests that the distribution of song popularity and user activity are extremely right-skewed. 75% of total number of users active no more than 152 times and 75% of the songs are played no more than 116 times.

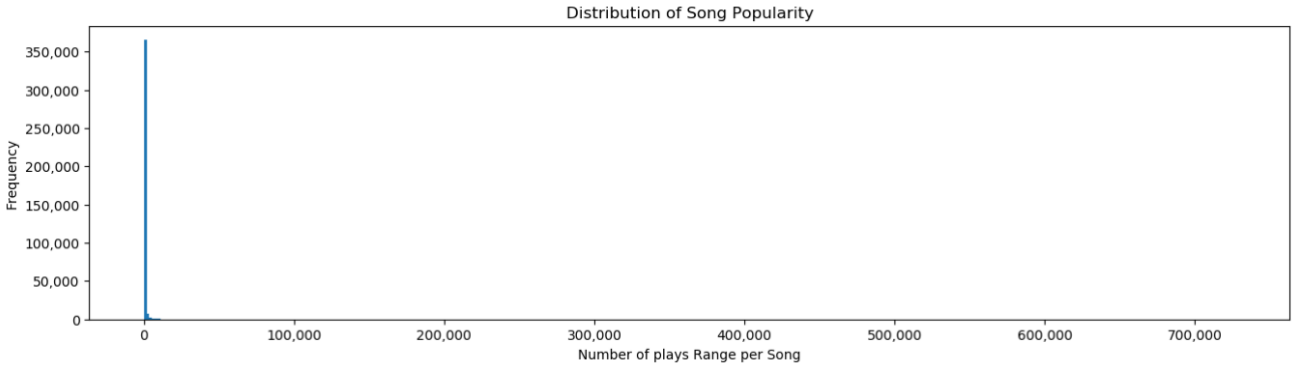


Figure 5. Distribution of Song Popularity

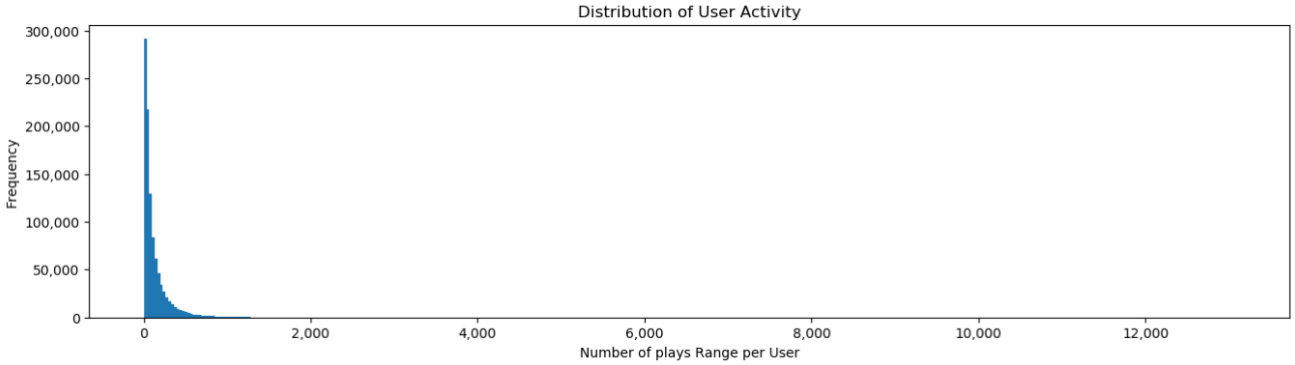


Figure 6. Distribution of User Activity

One important preprocessing step before building song recommendations is that we need to filter out the song that have been only played only a few times and users who have only listened to a few songs which do not provide much information to the overall dataset. First, we examine the descriptive statistics of number of song user listened to and the number of play of each song (table 21)

	Min	25%	50%	75%	100%
User song count	3.0	14.0	25.0	51.0	4,316.0
Song play count	1.0	6.0	29.0	119.0	726,885.0

Table 21. descriptive statistics of number of song user listened to and the number of play of each song.

We applied a filtering process to remove users with a song count less than M and songs with a play count less than N, where M and N are specific threshold values. The dataset was then split into a train/test set with a 70/30 ratio. It's important to note that when making predictions using the model, errors may occur if the users/songs in the test data are not present in the training data, as this is

inherent to collaborative filtering models. To address this, we removed those users/song from the test set before performing song recommendations.

Additionally, we ensured that the test set included at least 25% of the total play count of the entire dataset to maintain an adequate representation. The resulting clean datasets are referred to as table 22.

Thresholds	Train/Test's Total Observations	Total number of songs/users in test data not in train data need to be removed	proportion of total plays in test data in total dataset (%)
M = 13 N = 5	training: 30368734 test: 13011129 % remove: 5.2740	0	30.00
M = 7 N = 4	training: 31967933 test: 13701740 % remove: 0.2739	10	29.99
M = 6 N = 5	training: 31947120 test: 13695873 % remove: 0.3321	7	29.98
M = 7 N = 5	training: 31944397 test: 13690046 % remove: 0.3508	9	29.99
M = 8 N = 5	training: 31910403 test: 13676079 % remove: 0.4556	1	30.00
M = 9 N = 5	training: 31770100 test: 13619048 % remove: 0.8864	0	30.00
M = 11 N = 5	training: 31070413 test: 13313309 % remove: 3.0819	0	29.98
M = 13 N = 10	training: 30243273 test: 12960379 % remove: 5.6588	0	29.99
M = 15 N = 8	training: 29604881 test: 12688524 % remove: 7.6464	0	30.01
M = 13 N = 8	training: 30295311 test: 12979553 % remove: 5.5033	0	29.96

Table 22. Data filtering with different thresholds

Question 2. Train the collaborative filtering model.

We choose remove some observation with M = 13 and N = 5 which is before first quantile, and perform Alternating Least Squares (ALS) training with number of iterations is 5 and the regression evaluator. For the subset of users for song recommendation task, we extract top 10 most active users. We examine these users with their actual listented song and recommended song as table 23 below.

User_ID	Actual listened songs	Song recommendations
8	163850, 113, 11271, 16305, 25106, 44, 9, 350, 197, 104, 5327, 41933, 15, 5615, 114, 33850, 37837, 13, 45…	24, 15, 0, 7, 3, 6, 42, 2, 32, 10

9	1127, 89, 1068, 888, 75075, 72917, 17408, 23926, 3070, 445, 691, 109, 34, 3739, 121, 4293, 5706, 824…	24, 7, 20, 35, 6, 17, 23, 14, 46, 21
76	77835, 275, 80380, 91620, 963, 80232, 113256, 41274, 67288, 1151, 183090, 49151, 192242, 159676, 68429,…	91, 76, 0, 6, 162, 104, 130, 17, 10, 35
5168	83, 3, 11, 1755, 2704, 186228, 38, 1797, 1733, 124, 933, 2794, 49140, 36098, 20, 8642, 138, 162, 881, 787,…	3, 1, 13, 23, 38, 20, 41, 24, 83, 6
18092	23912, 26800, 8334, 104633, 82, 40305, 85044, 2999, 4355, 1092, 16899, 95670, 7245, 13031, 3548, 90457,…	52, 94, 90, 82, 189, 128, 0, 22, 387, 50
24067	1080, 15905, 4069, 114789, 22250, 24285, 31416, 52768, 58897, 29878, 64972, 59459, 507, 10741, 142319,…	89, 75, 622, 145, 13, 1, 92, 181, 83, 677
36877	2143, 526, 30313, 431, 15460, 2898, 77224, 151451, 112538, 27131, 90361, 26513, 34225, 14678, 38878,…	52, 1323, 22, 90, 94, 89, 134, 674, 42, 361
47605	321, 856, 558, 23680, 19827, 634, 68, 2351, 33, 631, 5, 1768, 1638, 13690, 669, 337, 17606, 1216,…	89, 24, 161, 364, 189, 90, 94, 1, 16, 7
208506	47805, 4508, 0, 66569, 507, 129, 50282, 8042, 214, 168, 406, 2212, 7385, 1032, 1241, 32839, 2881, 21535,…	4, 1, 3, 22, 28, 107, 13, 63, 81, 38
515326	18280, 2866, 50699, 8259, 14319, 21147, 34103, 75, 23774, 27604, 31999,…	1, 3, 4, 28, 13, 6, 22, 107, 38, 25
659627	158965, 10974, 676, 107600, 25736, 11193, 11914	89, 13, 622, 1, 38, 364, 83, 3, 100, 459
678615	16749, 180, 2139, 616, 4735, 39450.	92, 75, 12, 591, 125, 16, 17, 89, 10, 161

Table 23. Song recommendations for 10 random users.

It is evident that the recommendation system performance not so well when suggesting irrelevant songs to certain users. This issue can be attributed to the sparsity of data and the scalability of the dataset, as large user-song matrices can negatively impact the effectiveness of collaborative filtering. It suggests that the values chosen for N and M may not be sufficiently large to address this issue and enhance the overall effectiveness of the collaborative filtering approach.

Evaluating collaborative filtering model performance

Precision @ 10, NDCG @ 10, and Mean Average Precision (MAP) are commonly used metrics for evaluating collaborative filtering models in recommendation systems.

Precision @ 10: Precision is a metric that measures the proportion of correctly recommended items among the top-10 recommendations. It helps assess the accuracy and relevance of individual recommendations. There are some drawbacks of this metric: Precision @ 10 does not consider the order or ranking of the recommendations beyond the top 10. It may not provide a comprehensive view of the overall ranking quality or user satisfaction.

NDCG @ 10: Normalized Discounted Cumulative Gain (NDCG) is a metric that accounts for both relevance and ranking position of recommended items. NDCG @ 10 measures the quality of the top-10 recommendations, considering the relevance of the items and their positions in the ranking. It rewards both relevance and higher positions in the list. Its limitations are: NDCG @ 10 only considers the top 10 recommendations and does not provide a holistic evaluation of the entire recommendation list. It may not capture the overall performance for larger recommendation sets.

Mean Average Precision (MAP): MAP evaluates the average precision across all queries or predictions. It considers the precision at each position in the ranking and calculates the mean value. MAP provides a broader view of the overall performance of the recommendation system by taking into account the ranking quality and relevance across all predictions. However, MAP may not capture the variations in user preferences and satisfaction for different recommendations. It treats all queries or predictions equally, without considering their individual importance or impact.

Now we will use the trained model to generate song recommendations for test data, and examine the metrics Precision @ 10, NDCG @ 10, Mean Average Precision (MAP). We changing the value for M and N to pull out the best performance which produce the lowest values for those metrics. The results are shown as table 24.

Thresholds	Precision @ 10	NDCG @ 10	Mean Average Precision (MAP).
M = 6, N = 5	3.25%	4.33%	1.49%
M = 7, N = 4	3.24%	4.31%	1.49%
M = 7, N = 5	3.32%	4.43%	**1.51%**
M = 8, N = 5	3.28%	4.32%	1.46%
M = 9, N = 5	3.32%	4.36%	1.46%
M = 11, N = 5	3.49%	4.37%	1.37%
M = 13, N = 5	3.69%	**4.49%**	1.32%
M = 13, N = 8	3.66%	4.44%	1.32%
M = 13, N = 10	3.64%	4.40%	1.27%
M = 15, N = 8	**3.77%**	4.42%	1.22%

Table 24. Metric result for different thresholds.

Based on the given outcomes, the average Precision @ 10 score of approximately 3.5% suggests that only 3.5% of the top 10 predicted results are accurate. Similarly, the average NDCG @ 10 score of 4.4% indicates that the ranking quality of the top 10 results is not very high. Additionally, the average Mean Average Precision (MAP) score of 1.4% implies that the average precision across all queries or predictions is quite low.

When we increase M, Precision @ 10 and NDCG @ 10 metric values increase a little, meanwhile the MAP value is slowly decrease. Among the 10 different pairs of thresholds, the best overall system performance across multiple queries or predictions was achieved with M = 7 and N = 5, which resulted in the highest MAP value. On the other hand, for the highest accuracy and relevance of individual recommendations, the model performed best with M = 13 or 15 and N = 5, yielding the highest values for Precision @ 10 and NDCG @ 10.

An alternative approach to compare recommendation systems in real-world scenarios involves conducting A/B testing or online experiments. This method entails deploying different recommendation systems to actual users and measuring important performance indicators like click-through rates, conversion rates, or user engagement metrics. By analysing user behaviour and collecting feedback, it becomes possible to evaluate the actual impact and effectiveness of the recommendation systems in driving desired user actions. A/B testing allows for a more direct comparison and evaluation of recommendation systems in practical settings.