**IT2362-02 Predictive Modelling Assignment Report**

**BY:**

TEY JIA YING

**Table of Contents**

[**1.** **Introduction** 4](#_Toc94788045)

[**2.** **Data Understanding** 4](#_Toc94788046)

[**2.1** **Data Quantity** 4](#_Toc94788047)

[**2.2** **Data Types** 4](#_Toc94788048)

[**2.3** **Explanation of Dataset and Data Characteristics** 4](#_Toc94788049)

[**3.** **Data Preparation** 5](#_Toc94788050)

[**3.1** **Replacing Missing Values** 5](#_Toc94788051)

[**3.2** **Removing Duplicates** 5](#_Toc94788052)

[**3.3** **Converting Duration from Milliseconds to Minutes** 6](#_Toc94788053)

[**3.4** **Removing Outliers** 6](#_Toc94788054)

[**4.** **Exploratory Data Analysis** 7](#_Toc94788055)

[**4.1** **Descriptive Statistical Analysis** 7](#_Toc94788056)

[**4.1.1** **Statistical Summary of the Numerical Variables** 7](#_Toc94788057)

[**4.1.2** **Distribution of each Music Characteristics** 7](#_Toc94788058)

[**4.2** **Correlation Analysis** 8](#_Toc94788059)

[**4.2.1** **Music Correlation Bar Chart** 9](#_Toc94788060)

[**4.2.2** **Music Correlation Matrix** 9](#_Toc94788061)

[**4.3** **Song Class Analysis** 10](#_Toc94788062)

[**4.3.1** **Distribution and Proportion of Song Class** 10](#_Toc94788063)

[**4.4** **Further Analysis of Music Characteristics and Class** 10](#_Toc94788064)

[**4.4.1** **Median Music Characteristic across All Classes** 10](#_Toc94788065)

[**4.4.2** **Deeper Dive into Analysis between Music Characteristic and Class** 12](#_Toc94788066)

[**4.4.2.1**  **Deeper Dive into Analysis between Acousticness and Class** 13](#_Toc94788067)

[**4.4.2.2**  **Deeper Dive into Analysis between Energy and Class** 14](#_Toc94788068)

[**4.4.2.3**  **Deeper Dive into Analysis between Loudness and Class** 15](#_Toc94788069)

[**5.** **Data Modelling, Evaluation and Prediction** 16](#_Toc94788070)

[**5.1** **Preparing data for modelling** 16](#_Toc94788071)

[**5.1.1** **Dropping fields not used for classification** 16](#_Toc94788072)

[**5.1.2** **Split Input Features and label** 16](#_Toc94788073)

[**5.1.3** **Data Balancing** 17](#_Toc94788074)

[**5.1.4** **Feature Scaling** 17](#_Toc94788075)

[**5.1.5** **Feature Selection** 17](#_Toc94788076)

[**5.1.6** **Train Test Split** 18](#_Toc94788077)

[**5.2** **Modelling** 18](#_Toc94788078)

[**5.2.1** **Methodology** 18](#_Toc94788079)

[**5.2.2** **Models** 19](#_Toc94788080)

[**5.2.2.1** **Model I - Decision Tree Classifier** 19](#_Toc94788081)

[**5.2.2.2** **Model II – K-Neighbors Classifier** 22](#_Toc94788082)

[**5.2.2.3** **Model III – Random Forest Classifier** 24](#_Toc94788083)

[**5.2.2.4** **Model IV - XGBoost classifier** 26](#_Toc94788084)

[**5.2.2.5** **Model V – Gradient Boosting Classifier** 29](#_Toc94788085)

[**5.2.2.6** **Model VI – Multinomial NB** 30](#_Toc94788086)

[**5.2.2.7** **Model VII – SGD Classifier** 30](#_Toc94788087)

[**5.2.2.8** **Model VIII – One Vs Rest Classifier** 31](#_Toc94788088)

[**5.2.2.9** **Model IX – Neural Network Model** 32](#_Toc94788089)

[**6** **Comparison of models** 34](#_Toc94788090)

[**7** **Conclusion** 35](#_Toc94788091)

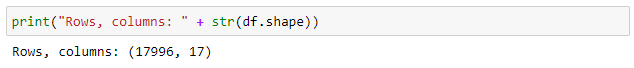
## **Introduction**

For this project, the aim is to classify a piece of music based on the characteristic or features of the music. I will be using a pre-selected dataset for this Data2021.csv .The file contains information about music and its classification. Each row in the file contains characteristics of a piece of music like the loudness, tempo etc. The music pieces are classified into 11 different classes (0 – 10). Each of these classes correspond to a type of music like Folk, Blue, HipHop, Metal Pop.

## **Data Understanding**

The first stage of any data analytics project is to have a good understanding of the data. Hence, I decided to understand the data quantity, data types and characteristics of each column to have a good gauge on what I could do with the data in terms of analysis, preparation, and modelling.

## **Data Quantity**



I used them. shape and found out that the dataset has 17996 rows and 17 columns which would be considered a good amount of data for analysis.

## **Data Types**

Table

Description automatically generated with medium confidence

Using the .dtypes function, we can see that most of the rows are of float or int type while only the columns containing Name are of object type.

## **Explanation of Dataset and Data Characteristics**

This dataset contains information about songs and its music characteristics.

After some research on each of the characteristics, I managed to get the definition of each characteristic.

A picture containing table

Description automatically generated

The other non-music characteristic columns are popularity, Class, Track Name, Artist Name.

* Popularity represents the amount of fame/support for each song.
* The target variable would be Class where it has a total 11 unique classes that represents different type of music such as Folk, Blue, HipHop, Metal Pop.
* The **2 other rows - Artist Name and Track Name** that is a unique identifier of each row.

## **Data Preparation**

After we have gotten a good understanding of our data, let us prepare the data for analysis and modelling. My methodology for this stage will be first, to replace missing values, then remove duplicates, followed by converting duration to minutes and lastly after all invalid/missing data is handled. I will be removing outliers

## **Replacing Missing Values**

Graphical user interface, text, application

Description automatically generated

I chose not to drop the null values and instead replace it with KNN imputation as total number of missing values for popularity, instrumentalness and key were ranging from 400 to 4300. If I were to drop all it will affect our dataset as we do not have much information on these 3 music characteristics to predict the class.

## **Removing Duplicates**

Let’s explore the number of duplicates in the whole dataset.

A picture containing text

Description automatically generated

Looks like there are no duplicates. However, as I scanned through the whole dataset, I realised that there are rows that have duplicated values across all columns except for Class. This would mean that there are 2 exactly songs having the same music characteristic but they of 2 different classes. Hence, let us drop data with similar values across all columns except for class

Text, application, email

Description automatically generated

Now, that we have dropped duplicates, we are left with 14838 rows.

## **Converting Duration from Milliseconds to Minutes**

After scanning through the dataset, I realised that the longest duration is minutes is 30. Hence, I will be converting the values above 30 from milliseconds to minutes.

Text, application

Description automatically generated

## **Removing Outliers**

Graphical user interface

Description automatically generated

In this stage, I first checked for the number of outliers using z score and realised that there were 1354 outliers which stands around 8% of the whole dataset. Hence, I decided to remove the outliers as there is enough data to spare.

Graphical user interface, table

Description automatically generated

Now that I have removed outliers, it will allow for a better model performance in the later stage.

## **Exploratory Data Analysis**

After we have cleaned the data, we will now move on to Exploratory Data Analysis (EDA) to explore and visualize some data to get some meaningful insights. For this stage, I will be exploring these few areas for analysis:

* [Descriptive Statistical Analysis](#_Descriptive_Statistical_Analysis)
* [Correlation Analysis](#_Correlation_Analysis)
* [Song Class Analysis](#_Song_Class_Analysis)
* [Further Analysis of Music Characteristic and Class](#_Further_Analysis_of)

## **Descriptive Statistical Analysis**

## **Statistical Summary of the Numerical Variables**

I used the describe () function once again to look at the statistical summary of the numerical variables.

Table

Description automatically generated Table

Description automatically generated

From this data frame, we may not be able to visualize at one glance the distribution of all characteristics. Hence, I have decided to plot the distribution of each characteristic below

## **Distribution of each Music Characteristics**

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, bar chart, histogram

Description automatically generated

**Insights:**

|  |  |
| --- | --- |
| **Distribution** | **Music Characteristics** |
| Symmetrical | 1. Danceability 2. Valence |
| Left skewed | 1. Energy 2. Loudness |
| Right skewed | 1. Speechiness 2. Liveness 3. Acousticness 4. Instrumentalness  5. Duration in min |
| Multimodal | 1. Key  2. Tempo |
| Have only 1 or 2 bars in their distribution | 1. Mode 2. Time Signature |

## **Correlation Analysis**

Next, I did a Correlation Analysis to better understand the relationship between the 14 key factors affecting the class of the music

## **Music Correlation Bar Chart**

Chart, histogram, waterfall chart

Description automatically generated

## **Music Correlation Matrix**

**Chart, waterfall chart

Description automatically generated**

**Insights:**

**Correlation between music characteristics and class**  
Overall correlation between the 13 variables and the Class is not very strong. However, I have identified four variables which had significant correlation with the Class

1. Acousticness Vs Class(-0.3)  
2. Energy Vs Class(0.28)  
3. Loudness Vs Class(0.24)

**Correlation among music characteristics**

High Correlation exist among these music characteristics

1. Loudness Vs Energy (0.76)
2. Acousticness Vs Energy (-0.74)
3. Acousticness Vs Loudness (-0.59)

Hence, for the modelling stage, I will be doing feature selection to avoid multicollinearity

## **Song Class Analysis**

Next, I will be analysing the class of the music to get a better understanding of it.

## **Distribution and Proportion of Song Class**

Chart, pie chart

Description automatically generatedChart, line chart, histogram

Description automatically generated

As we can see from the above pie chart, Class 10 stands the highest percentage of 28.2% followed by class 9 then 6. The classes with the lower data examples are classes 4,3,7,0. They have around 2% to 4% of data examples which is around 360 to 540.

There is uneven distribution between 11 classes where Class 10 hits the highest percentage of 28.2%. There are many minority classes such as class 7,3,4,0 where their population is less than 5%. Hence, during modelling stage, I will be balancing the data.

## **Further Analysis of Music Characteristics and Class**

## **Median Music Characteristic across All Classes**

Graphical user interface, text, application

Description automatically generated

Next, I create a plot\_genre\_horizontal\_bar () function to group the classes and sort it by the median value of the music characteristic. I then plotted each music characteristic across each class of songs out. Take note that I normalized the value of the music characteristic for fairer comparison.

***\*\*Graphs will be displayed first followed by insights.***

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Chart

Description automatically generatedChart, bar chart

Description automatically generated

**Insights on music characteristics across different music genres/classes:**

Class with highest danceability: **Class 5**

Class with highest energy: **Class 8**

Class with highest volume(loudness): **Class 8**

Class with highest speechiness: **Class 5**

Class with highest acousticness**: Class 7**

Class with highest liveness: **Class 8**

Class with highest valence: **Class 3**

Class with highest tempo: **Class 8**

Class with highest duration: **Class 3**

Class with highest instrumentalness: **Class 7**

**Further Interesting Insights:**

1. Songs of class 8 have the highest value for most music characteristics.
2. For instrumentalness in each class, all classes except for class 7 have near to 0.0 instrumentalness.

## **Deeper Dive into Analysis between Music Characteristic and Class**

In this section, we will only focus on the 3 most significant variables(Acousticness, Energy and Loudness ) as mentioned in the correlation matrix above. I will dive deeper to analyse the relationship between each of the 3 factors and the Class to produce some insights from it.

## **4.4.2.1 Deeper Dive into Analysis between Acousticness and Class**

Chart

Description automatically generated with medium confidence

***Insights:***

a) **Distribution Chart of acousticness:**  
The distribution chart of **acousticness** is right skewed which means that the quantity of acousticness is quite less in this dataset. The quantity of acousticness ranges between 0.0 to 1.0 g/dm^3 where the distribution mainly clusters around 0.0 to 0.2.

b) **Distribution Chart of class:**  
We can see that the chart is left-skewed where most of the class is class 10.

c) **Boxplot for acousticness:**  
From the boxplot, we can see that Classes 0,3,7 has a higher value of acousticness where the median value of acousticness is above 0.6. Whereas for classes 1，2，4，5，6，8，9 and 10, song in this class have a lower value of acousticness where median value is above 0.4.

d) **Joint Plot for acousticness and class:**  
We have combined the two distribution charts together to see their relationship. We learned that there is a negative relationship between acousticness and class as the orange line shows a negative linear graph. This evidences the results from the boxplot where the increase in acousticness results in a decrease in class number. The graph is quite steeped up which shows that the relationship between acousticness and class is quite strong.

## **4.4.2.2 Deeper Dive into Analysis between Energy and Class**

Chart

Description automatically generated

***Insights:***

a) **Distribution Chart of Energy:**  
The distribution chart of energy is very left skewed which means that the quantity of energy is quite a lot in this dataset. The quantity of energy ranges between 0.0 to 1.0 where the distribution is higher when value of energy increases

b) **Distribution Chart of class:**  
We can see that the chart is left-skewed where most of the class is class 10.

c) **Boxplot for energy:**  
From the boxplot, we can see that for most classes, their energy level is quite high for most classes the median value of energy is 0.5 and above. For Class 7, the energy level is very low as compared to other Classes with the median value of energy at 0.1~.

d) **Joint Plot for energy and class:**  
We have combined the two distribution charts together to see their relationship. We learned that there is a positive relationship between energy and class as the orange line shows a positive linear graph. This evidences the results from the boxplot where the increase in energy results in an increase in class number. The graph is quite steeped up which shows that the relationship between energy and class is quite strong.

## **4.4.2.3 Deeper Dive into Analysis between Loudness and Class**

Chart

Description automatically generated

***Insights:***

a) **Distribution Chart of loudness:**  
The distribution chart of loudness is very left-skewed which means that the quantity of loudness is quite a lot in this dataset. The quantity of loudness is higher when value is higher (around -10 to -5）。

b) **Distribution Chart of class:**  
We can see that the chart is left-skewed where most of the class is class 10.

c) **Boxplot for loudness:**  
From the boxplot, we can see that for most classes, their loudness level is quite high for most classes the median value of loudness is -10 and above. For Class 7, the loudness level is very low as compared to other Classes with the median value of loudness at -17~.

d) **Joint Plot for loudness and class:**  
We have combined the two distribution charts together to see their relationship. We learned that there is a positive relationship between loudness and class as the orange line shows a positive linear graph. This evidences the results from the boxplot where the increase in loudness results in an increase in class number. The graph is quite steeped up which shows that the relationship between loudness and class is quite strong.

## **Data Modelling, Evaluation and Prediction**

## **Preparing data for modelling**

For the preparation of data for modelling, my methodology would be:

1. Drop fields that are not used for classification
2. Split Input Features and Label
3. Data Balancing
4. Feature Scaling
5. Feature Selection
6. Train Test Split

*Note: I have tried another* ***failed approach*** *which is to:*

1. Drop fields that are not used for classification
2. Split Input Features and Label
3. Train Test Split
4. Data Balancing
5. Feature Scaling
6. Feature Selection

In this failed approach, I train test split the data before balancing and scaling it. This would mean that only the train data is balanced and scaled. Test data was not. The model could not perform well as the unseen data or test data was in a different format from the training data. Hence, this approach turns out to give minimal accuracy and precision of 10% to 40%.

After seeking out some online resource and consults, I will be following the first methodology that I mentioned instead of the failed one.

## **Dropping fields not used for classification**

Let us first drop columns that are not definitely involved in the prediction of class such as popularity, song name and track name. Popularity is not a music characteristic and should not be used for predicting classes. Whereas the song name and track name is just a unique identifier for each row.

Graphical user interface, application, table

Description automatically generated

## **Split Input Features and label**

Text

Description automatically generated with medium confidence

## **Data Balancing**

I then balanced the whole dataset (X,y) using imblearn.over\_sampling.

Graphical user interface, text, application, email

Description automatically generated

## **Feature Scaling**

I then did feature scaling using sklearn.preprocessing.MinMaxScaler () method to scale the data/normalize the data to a range of 0 to 1 for fairer comparison.

Noticed that I only feature scaled non-categorical variable. For encoded categorical variable such as "mode”, “time\_signature" and "key", I did not scale it because it is already encoded. (E.g., Key=B# is 9)

Text

Description automatically generated

## **Feature Selection**

Note that for feature selection, I will be carrying it out in my first model(Decision Tree) as I will be able to compare the results before and after selecting the features needed to increase model performances. I would need to develop a model first to carry out this process.

\*\* *Refer to 5.2.2.1- Feature selection stage*

## **Train Test Split**

Lastly, after preparing the data for modelling, I then performed train\_test\_split

Text

Description automatically generated

## **Modelling**

From our analysis above, we now have a better understanding on the relationship between each music's characteristics and classes.

## **Methodology**

* + - 1. **Selecting Classification Models**

From our analysis above, we now have a better understanding on the relationship between each music's characteristics and classes. As our objective is to better understand what factors affect the class (type of music), we will need to use classification model.

There are many types of classification model:

1. [Decision Tree Classifier](http://localhost:8888/notebooks/Downloads/PM_Asssignment/Assignment/202704D_PM_Project4.ipynb#10)
2. [K-Neighbors Classifier](http://localhost:8888/notebooks/Downloads/PM_Asssignment/Assignment/202704D_PM_Project4.ipynb#11)
3. [Random Forest Classifier](http://localhost:8888/notebooks/Downloads/PM_Asssignment/Assignment/202704D_PM_Project4.ipynb#12)
4. [XGBoost Classifier](http://localhost:8888/notebooks/Downloads/PM_Asssignment/Assignment/202704D_PM_Project4.ipynb#14)
5. [Gradient Boosting Classifier](http://localhost:8888/notebooks/Downloads/PM_Asssignment/Assignment/202704D_PM_Project4.ipynb#13)
6. [Multinomial Naive Bayes](http://localhost:8888/notebooks/Downloads/PM_Asssignment/Assignment/202704D_PM_Project4.ipynb#15)
7. [Stochastic Gradient Descent(SGD) Classifier](http://localhost:8888/notebooks/Downloads/PM_Asssignment/Assignment/202704D_PM_Project4.ipynb#16)
8. [One-Vs-Rest Classifier](http://localhost:8888/notebooks/Downloads/PM_Asssignment/Assignment/202704D_PM_Project4.ipynb#17)
9. [Neural Network](http://localhost:8888/notebooks/Downloads/PM_Asssignment/Assignment/202704D_PM_Project4.ipynb#18)
10. *Logistic Regression (only for binary class classification)*
11. *Support vector Machine (only for binary class classification)*

However, I will omitting **Logistic Regression and SVM** as both can only are for binary classification but in this case, we have 11 classes.

**5.2.1.2 How will I be carrying out this process?**

1. Try out each model
2. Fine tune parameters to give the BEST results for each model
3. Compare the best results of each model
4. Derive the model that gives the BEST results out of all models

**5.2.1.3 Metrics used** - **Why I use cross validation?**

I will be using cross validation rather than classification report of precision, recall, f1-score and accuracy, as it allows a better representation assessment of the model performance as it test each and every portion of the dataset for each model. This is because our dataset. Also, given that the dataset is small, it is better to get a good representative of each portion of our dataset to get a good and reliable test result.

**5.2.1.4 Metrics used** - **What area of metric will I be focusing on?**

However, I will be focusing more on precision in this project. This is because we are not targeting on health-related classification or prediction where false negatives are important. In this case, more false negatives are not as costly. Hence, we be focusing on false positive, preciseness of our classification.

## **5.2.2 Models**

## **5.2.2.1 Model I - Decision Tree Classifier**

The first model I will be using would be using is decision tree classifier.

**Step A: Building Model - Decision Tree**

Graphical user interface, text, application

Description automatically generated

I imported the DecisionTreeClassifier function from the sklearn.tree library. Take note that this model I build includes no parameters, only a random state.

**Step B: Model Evaluation**

We can see that results for **classification report performance are lower than our cross-validation performance**. Hence, in the next few models, **I will be using cross validation**as it allows me to test and train every portion of the dataset. Also, given that the dataset is small, it is better to get a good representative of each portion of our dataset to get a good and reliable test result.

Graphical user interface, text, application

Description automatically generated

I then performed cross validation of 10-folds to get a reliable precision.

At this stage, the model performance is at an **average performance**. We can see that accuracy is 0.64, precision is 0.63, recall is 0.64 and F1 is 0.63.

**Feature Selection**

I will be using mlxtend.feature\_selection to do **forward feature selection** to find the best subset of features that gives highest precision.

Text

Description automatically generated

Looking at the results, **subset 12** which contains 'danceability', 'energy', 'loudness', 'acousticness', ' speechiness ', 'valence', 'tempo', 'duration\_in\_min', 'time\_signature', 'instrumentalness', 'key' **gives highest precision of 0.6339**.

Hence, we will be using this subset of features to build a new decision tree model and do cross validation to find out the performance of the model.

Text

Description automatically generated

Looks like the results have improved after using subset 12 of features. It increase from 0.63 range to 0.64 range of each of the performance metric.  
Hence, we will be using this **subset of features consisting of 'danceability', 'energy', 'loudness', 'acousticness', ' speechiness ', 'valence', 'tempo', 'duration\_in\_min', 'time\_signature', 'instrumentalness', 'key' for the rest of our models.**[**¶**](http://localhost:8888/notebooks/Downloads/PM_Asssignment/Assignment/202704D_PM_Project4.ipynb#-Hence,-we-will-be-using-this-subset-of-features-consisting-of-'danceability','energy','loudness','acousticness','speechiness','valence','duration_in-min',-'time_signature','instrumentalness'-for-the-rest-of-our-models.-)

**Step C: Tuning Parameters**

**How will I be tuning the parameters?**

I will be using **GridSearchCV to do hyper-parameter to find the best combination of parameters that gives highest precision**

**Hyper-parameter tuning – GridSearchCV**

Now, I will be using Grid Search CV to find the best combination of parameters and the corresponding value.

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

We see that by using a combination of parameters [**criterion='entropy', max\_depth=22, random\_state=42**], the overall results did improve.

**Step D: Model Insights**

Now let me do a comparison of all the decision tree models I have fine-tuned.

A picture containing application

Description automatically generated

From this table we can see that we can use the last model which is**DecisionTreeClassifier(criterion='entropy', max\_depth=22, random\_state=42)** as it **gives the higher precision, accuracy, recall and f1-score.**

**Conclusion: Highest precision of tuned decision tree is 64%.**

**Step E: Predictions**

Now, that we have derived the best decision tree model, we can use it to do some predictions on unseen data.

Text

Description automatically generated

## **5.2.2.2 Model II – K-Neighbors Classifier**

The second model I will be using would be using is K-Neighbors classifier.

**Step A: Building Model – K-Neighbors Classifier**

A picture containing application

Description automatically generated

I imported the KNeighborsClassifier function from the sklearn.neighbor’s library. Take note that this model I build includes no parameters, only a random state.

**Step B: Model Evaluation**

Graphical user interface, text, application

Description automatically generated

I then did model evaluation by performing cross validation of 10-folds to get a reliable accuracy. At this stage, the model performance is performing good with precision at 0.64 and accuracy and recall at 0.65.

**Step C: Tuning Parameters**

Now, I will be tuning the parameters using GridSearchCV.

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Looks like the model performance has improved a lot. We see that by using a combination of parameters [leaf\_size=25, n\_neighbors=1, p=1] to give the **best precision of 0.79,** accuracy of 0.79, recall of 0.79 and f1-score=0.79.

**Step D: Model Insights**

Now let me do a comparison of all the K-Neighbors Classifier I have fine-tuned.

Text

Description automatically generated with low confidence

From this table we can see that we can use the last model which consist of**KNeighborsClassifier (**leaf\_size=25, n\_neighbors=1, p=1**)** as it **gives best precision of 0.79, accuracy of 0.79, recall of 00.79 and f1-score=0.79.**

**Conclusion: Highest precision of tuned K-Neighbors Classifier is 79%.**

**Step E: Predictions**

Now, that we have derived the best K-Neighbors Classifier model, we can use it to do some predictions on unseen data.

A picture containing text

Description automatically generated

## **Model III – Random Forest Classifier**

The third model I will be using would be using is Random Forest Classifier

**Step A: Building Model – Random Forest Classifier**

Word

Description automatically generated

I imported the RandomForestClassifier () function from the sklearn.ensemble library. Take note that this model I build includes no parameters.

**Step B: Model Evaluation**

Text

Description automatically generated

I then did model evaluation by performing cross validation of 10-folds to get a reliable accuracy. At this stage, the model performance is performing good with precision at 0.79 and accuracy and recall at 0.80.

**Step C: Tuning Parameters**

Now, I will be tuning the parameters using GridSearchCV.

Graphical user interface, text, email

Description automatically generated

Looks like the model performance has by a little. We see that by using a combination of parameters {'max\_features': 'sqrt', 'n\_estimators': 1000}

to give the best precision of 0.80, accuracy and recall of 0.80

**Step D: Model Insights**

Now let me do a comparison of all the Random Forest Classifiers I have fine-tuned.

A picture containing graphical user interface

Description automatically generated

From this table we can see that we can use the last model which consist of Random Forest Classifier (max\_features='sqrt', n\_estimators=1000) as it gives best precision of 0.80, accuracy and recall of 0.80.

**Conclusion: Highest precision of tuned Random Forest Classifier is 80%.**

**Step E: Predictions**

Text

Description automatically generated with low confidence

## **Model IV - XGBoost classifier**

The fifth model I will be using would be using is XGBoost Classifier.

**Step A: Building Model – XGBoost Classifier**

Text

Description automatically generated

I imported the XGBClassifier function from the xgboost library. Take note that this model I build includes no parameters.

**Step B: Model Evaluation**

Text

Description automatically generated with medium confidence

I then did model evaluation by performing cross validation of 10-folds to get a reliable result.

At this stage, the model performance is quite good. We can see that precision is 0.73, accuracy is 0.73, recall is 0.73 and f1-score is 0.73.

**Step C: Tuning Parameters**

I will just be using grid search CV to do hyper-parameter tuning.

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Text

Description automatically generated**

Looks like the model performance has improved. We see that by using a combination of parameters

**[base\_score=0.5, booster='gbtree', colsample\_bylevel=1, colsample\_bynode=1, colsample\_bytree=0.7, enable\_categorical=False, gamma=0.0, gpu\_id=-1, importance\_type=None, interaction\_constraints='', learning\_rate=0.2, max\_delta\_step=0, max\_depth=15, min\_child\_weight=1, missing=nan, monotone\_constraints='()', n\_estimators=100, n\_jobs=16, num\_parallel\_tree=1, objective='multi:softprob', predictor='auto', random\_state=0, reg\_alpha=0, reg\_lambda=1, scale\_pos\_weight=None, subsample=1, tree\_method='exact', validate\_parameters=1, verbosity=None]**

to give the best precision of 0.81, accuracy of 0.81, recall of 0.81 and f1-score of 0.81.

**Step D: Model Insights**

Now let me do a comparison of all the XGBoost Classifier I have fine-tuned.

Scatter chart

Description automatically generated with medium confidence

From this table we can see that we can use the tuned model which consist **of XGBClassifier(base\_score=0.5, booster='gbtree', colsample\_bylevel=1, colsample\_bynode=1, colsample\_bytree=0.7, enable\_categorical=False, gamma=0.0, gpu\_id=-1, importance\_type=None, interaction\_constraints='', learning\_rate=0.2, max\_delta\_step=0, max\_depth=15, min\_child\_weight=1, missing=nan, monotone\_constraints='()', n\_estimators=100, n\_jobs=16, num\_parallel\_tree=1, objective='multi:softprob', predictor='auto', random\_state=0, reg\_alpha=0, reg\_lambda=1, scale\_pos\_weight=None, subsample=1, tree\_method='exact', validate\_parameters=1, verbosity=None)** as it **gives the** best precision of 0.81, accuracy of 0.81, recall of 0.81 and f1-score of 0.81.

**Conclusion: Highest precision of tuned XGBoost Classifier is 81%.**

**Step E: Predictions**

Now, that we have derived the XGBoost Classifier model, we can use it to do some predictions on unseen data.

Text, application

Description automatically generated

## **Model V – Gradient Boosting Classifier**

The fourth model I will be using would be using is Gradient Boosting Classifier.

**Step A: Building Model – Gradient Boosting Classifier**

Graphical user interface, application, Word

Description automatically generated

I imported the GradientBoostingClassifier () function from the sklearn.ensemble library. Take note that this model I build includes no parameters.

**Step B: Model Evaluation**

Graphical user interface, application

Description automatically generated

I then did model evaluation by performing cross validation of 10-folds to get a reliable accuracy. At this stage, the model performance is performing good with precision at 0.58 and accuracy and recall at 0.59.

**Note:** I will not be tuning this classifier as the base model already has a much lower performance than the base model of other algorithms. Hence, even with tuning, it may only increase a little bit but not enough to hit the 70% range of precision/accuracy/recall. Also, since accuracy is low, I will not be using this model to do any prediction.

**Conclusion: Highest precision of tuned Gradient Boosting Classifier is 58%.**

## **Model VI – Multinomial NB**

The sixth model I will be using would be using is Multinomial NB Model.

**Step A: Building Model – Multinomial NB**

Word

Description automatically generated

I imported the MultinomialNB () function from the sklearn.naive\_bayes library. Take note that this model I build includes no parameters, only a random state.

**Step B: Model Evaluation**

Text

Description automatically generated

I then did model evaluation by performing cross validation of 10-folds to get a reliable accuracy. At this stage, the model performance is performing below average as compared with previous model with precision at 0.37 and accuracy and recall at 0.41.

**Note:** I will not be tuning this classifier as the base model already has a much lower performance *(30 to 40% range)* than the base model of other algorithms. Hence, even with tuning, it may only increase a little bit but not enough to hit the 70% range of precision/accuracy/recall. Also, since accuracy is low, I will not be using this model to do any predictions.

**Conclusion: Highest precision of tuned Multinomial Naive Bayes Classifier is 38%.**

## **Model VII – SGD Classifier**

The seventh model I will be using would be using is SGD Classifier Model.

**Step A: Building Model –SGD Classifier**

Word

Description automatically generated with medium confidence

I imported the SGDClassifier () function from the sklearn. linear\_model library. Take note that this model I build includes no parameters.

**Step B: Model Evaluation**

Text

Description automatically generated

I then did model evaluation by performing cross validation of 10-folds to get a reliable accuracy. At this stage, the model performance is performing below average as compared with previous model with precision at 0.44 and accuracy and recall at 0.42.

**Note:** I will not be tuning this classifier as the base model already has a much lower performance *(40% range)* than the base model of other algorithms. Hence, even with tuning, it may only increase a little bit but not enough to hit the 70% range of precision/accuracy/recall. Also, since accuracy is low, I will not be using this model to do any predictions

**Conclusion: Highest precision of SGD Classifier is 44%.**

## **Model VIII – One Vs Rest Classifier**

The seventh model I will be using would be using is **One Vs Rest Classifier** Model.

**Step A: Building Model – One Vs Rest Classifier**

Graphical user interface, text, application

Description automatically generated

I imported the OneVsRestClassifier () function from the sklearn.multiclass library. Take note that this model I build includes no parameters except for an estimator.

**Step B: Model Evaluation**

Graphical user interface, text, application, email

Description automatically generated

I then did model evaluation by performing cross validation of 10-folds to get a reliable accuracy. At this stage, the model performance is performing below average as compared with previous model with precision at 0.42 and accuracy and recall at 0.47.

**Note:** I will not be tuning this classifier as the base model already has a much lower performance *(40% range)* than the base model of other algorithms. Hence, even with tuning, it may only increase a little bit but not enough to hit the 70% range of precision/accuracy/recall

**Conclusion: Highest precision of One-Vs-Rest Classifier is 42%.**

## **Model IX – Neural Network Model**

**Step A: Preparing Data for neural network model**

Different from machine learning model, for my neural network model requires the encoded class column to be converted in to dummy variables. Hence, I encoded and converted the y\_train and y\_test into 0 and 1s.

**Text

Description automatically generated with low confidence**

**Step B: Building and Tuning Model – Deep Neural Network**

Next, I started to build my deep neural network. I decided to build a sequential model.

I created 4 dense layer with relu activation function each. The first one being the input layer and followed by 2 dense layers being the middle layers. For the input and middle layers I decided to give each layer 512 neurons. For the last dense layer, it is output layer. For last layer, since there were 11 classes of songs, the number of output neurons would be set at 11.

**Tuning Process:**

At first, I set the neurons at the range of 16 to 32 but I turned out to give a lower accuracy. Also, at first, there was only 1 layer. However, I realised the more layers I add the higher performance it gave. However, I stopped at the 4th layer as the model performance was starting to decrease. In my training process, I also added a early stopping call back as it is a good method that allows me to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out validation dataset.  I also tested with differen batch size and realized the optimal batch size was 10.

Text

Description automatically generated

Graphical user interface, table

Description automatically generated

**Step C: Model Evaluation**

Graphical user interface, text

Description automatically generated

Chart

Description automatically generated Chart, line chart

Description automatically generated

I then printed the classification report and realised that the performance is quite good with precision, accuracy, recall and f1-score at 0.70~. Also, looks like the model accuracy and loss for training and testing performance is quite close which is a good sign of the model performing well.

**Note:** Based on my research, neural network models take millions of data to train. Although the model performance is good, it may be better suited for applications where the dataset is sufficiently large. In this project, it is more exploratory in nature and to gain a better understanding of neural network models.

**Conclusion: Highest precision of Tuned Deep Neural Network Model is 70%**

## **Comparison of models**

Now, let’s compared to see which fine-tuned model performed the best. I have picked the best performing model of each algorithm.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fine- Tuned model of algorithm** | **Model Information** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| Decision Tree | DecisionTreeClassifier(criterion='entropy', max\_depth=22, random\_state=42) | 0.65 | 0.64 | 0.65 | 0.64 |
| K-Neighbors Classifier | KNeighborsClassifier(leaf\_size=4, n\_neighbors=1, p=1) | 0.79 | 0.79 | 0.79 | 0.79 |
| Random Forest Classifier | RandomForestClassifier(max\_features='log2', n\_estimators=1000) | 0.80 | 0.80 | 0.80 | 0.80 |
| XGBoost Classifier | XGBClassifier(base\_score=0.5, booster='gbtree', colsample\_bylevel=1, colsample\_bynode=1, colsample\_bytree=0.7, enable\_categorical=False, gamma=0.0, gpu\_id=-1, importance\_type=None, interaction\_constraints='', learning\_rate=0.2, max\_delta\_step=0, max\_depth=15, min\_child\_weight=1, missing=nan, monotone\_constraints='()', n\_estimators=100, n\_jobs=16, num\_parallel\_tree=1, objective='multi:softprob', predictor='auto', random\_state=0, reg\_alpha=0, reg\_lambda=1, scale\_pos\_weight=None, subsample=1, tree\_method='exact', validate\_parameters=1, verbosity=None) | **0.81** | **0.81** | **0.81** | **0.81** |
| Gradient Boosting Classifier | GradientBoostingClassifier(learning\_rate= 0.001, max\_depth=9, n\_estimators=1000, subsample=0.5) | 0.59 | 0.58 | 0.59 | 0.57 |
| Multinomial Naive Bayes | MultinomialNB() | 0.41 | 0.38 | 0.41 | 0.38 |
| SGD Classifier | SGDClassifier() | 0.42 | 0.44 | 0.42 | 0.36 |
| One-Vs-Rest Classifier | OneVsRestClassifier(estimator=LinearSVC(random\_state=0)) | 0.47 | 0.42 | 0.42 | 0.36 |
| Neural Network | (Refer to model summary)  Graphical user interface, table  Description automatically generated | 0.70 | 0.70 | 0.71 | 0.70 |

**Conclusion: The best model to be used is XGBoost Classifier with the highest accuracy of 81%, precision of 81%, recall 0f 81% and f1-score of 81%.**

## **Conclusion**

For this report, I have gone through the whole data pipeline from data understanding, preparation, exploratory data analysis, modelling, evaluation, prediction and finally comparison of models. To optimize the performance of the model, I have tried many methodologies and switched the order of the data preparations and modelling preparation steps. The insights derived from a wide variety of models also contributed greatly to the overall objective, which is to classify songs by music characteristics. It is evident that different models with different parameters can indeed affect model performance. Additionally, the more models tested, the higher probability of finding a good model for our predictions. I hope this report will provide great value to you and assist you to find the best model that will derive the most accurate and precise predictions.