Investigating the performance of a Decision Tree Classifier and a Logistic Regressor on a Year of Music Release Prediction Dataset

Data Mining Project

Classifying Musical Decades 1922-2011

Clara Maine s1032005
Manthan Gadhia s1032698

Abstract

When talking about popular music of the fairly recent past, most people tend to intuitively classify music by the decade in which it was released ("50s Music", "80s Music", etc). The use of these terms tends to evoke a loose understanding of the style and genre of the music being referred to. This seems to indicate a general agreement that popular music from each of these decades shares some recognizable qualities which, we hypothesize, can be quantified and grouped together. This report discusses and explains in detail the approaches taken to find a classification model in order to classify songs based only on their auditory characteristics. Two separate classification models were implemented, one using a Decision Tree and one using logistic regression, which are then compared using ROC curves.

1 Introduction

We have chosen to do "Problem".

Human beings noticeably have a tendency to group music into eras that span a decade i.e. music will often be talked about as belonging to the "20s" or the "50s", etc. In this report then, we aim to investigate if there exist enough quantifiable similarities (or differences) amongst songs which human beings categorise into eras such that these categories can be recreated computationally.

The procedure being used to approach this problem consists of using two separate classification algorithms (firstly a Decision Tree Classifier, and secondly a logistic regression classifier) which will be fit on the dataset. From there, their performances will be analysed with the hope that classification of music based on it's sound characteristics into specific eras is feasible.

The dataset being used for this project is called *YearPredictionMSD* based off of the very popular *Million Song Dataset*. It has been cleaned up and curated specifically to be able to perform year prediction on the songs based on it's average timbre values. The *YearPredictionMSD* consists of 515345 entries, 90 attributes, and originally it consisted of 1 label ranging from 1922 - 2011 (inclusive) — the year of release of the given song. However, in order to fit the question being investigated in this report, this label was replaced with another titled *'Decade'*, and this also served the purpose of making the models have more generalised training data, avoiding overfitting. In the classification tasks being carried out, this *'Decade'* is the label being predicted.

Every algorithmic implementation as well as all data analysis discussed in this report was done using Python 3 in a Python Notebook (.ipynb), and relevant libraries available for this language were used (such as `sci-kit learn, NumPy, MatPlotLib, Pandas, Seaborn,` etc.).

2 Methods

The dataset being used to carry out this investigation, the *YearPredictionMSD*, as described in the previous section, consists of 515345 entries, 90 attributes, and 1 label.

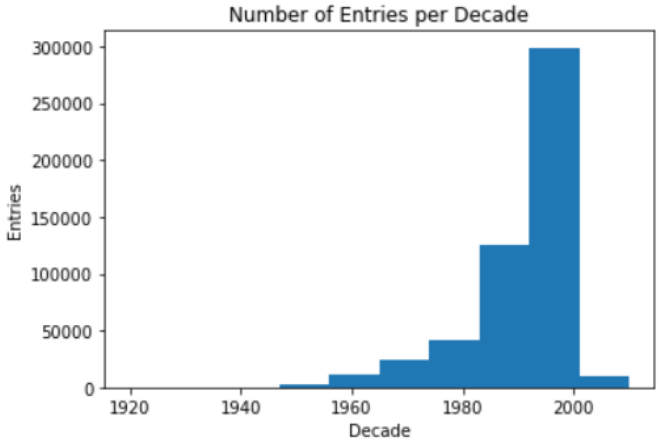
Of the 90 attributes, the first 12 ([Timbre1, Timbre2, ..., Timbre12]) correspond to the averages of the first 12 principle components of the timbre of each song. The following 78 attributes consist of the covariances of each average timbre component with every other average timbre component; this would technically make 144 attributes ($12 \times 12 = 144$), However $Covariance_{1,2} = Covariance_{2,1}$, and therefore removing the duplicates leaves half of the covariances: 78. Of the 515345 entries, each represents a distinct song released at some point of time between (and including) the years 1922 and 2011, however these are not sorted in any noticeable pattern (e.g. chronologically based on year of release). The custom label, *Decade*, uses the original *Year* label, and assigns the decade that the year belonged to to this new label. Distinctions between decades are made when the year switches from one ending with a 9 to one ending with a 0, i.e. every entry with a year of release in the inclusive range 1960-69 will be assigned the decade 1960, and a similar procedure is used for every other decade.

2.1 Pre-processing

Firstly, some minor chekcs were conducted to make sure that the dataset was orderly, that there weren't any incomplete records, that there weren't any attributes which might cause problems for analysis (by being an incompatible type for example), and so on. No issues of this kind were found.

Following this, when carrying out a preliminary inspection of the dataset, some characteristics were noted to be potential hindrances, which needed to be normalised in order to be able to obtain

relevant results from any analyses to be carried out. The first of these was the frequency distribution of entries available for each training label:




```
Most Frequent Decade: 2000 accounts for 58.02 percent of data.
2nd Most Frequent Decade: 1990 accounts for 24.2 percent of data.
Together, these two decades account for 82.22 percent of data.
```

Figure 1: The distribution of the number of entries in the original dataset per decade

To combat this affecting the models being trained, it was beneficial to the in-built feature offered by most sci-kit learn implementations which balances the training data such that the same proportion of training data is selected from each label. The instantiations of the models being called were structured as follows:

```
clf = tree.DecisionTreeClassifier(min_samples_split = 100, max_depth=d, class_weight='balanced')
reg_clf = LogisticRegression(solver='lbfgs', class_weight='balanced')
```

The next step was to study the overall distribution of the labels throughout the dataset, and they were seen to not be in chronological order. This meant that when splitting the dataset into a training and testing sample, a comparable distribution of entries of each class would be preferable. In case this distribution was not semi-uniform in it's random ordering, there would be risk of ending up with a training subset that is not proportionately representative of the whole dataset that is being studied. To investigate this further, a scatter-plot of the decade of each entry against it's index within the dataset; this was plotted with 25000-entry moving average with the aim of seeing if there were any largely skewed portions within the dataset:

From the figure above, it is clear that the spread of entries throughout the dataset is fairly uniform in its randomness, implying that there should not be any test-train splits that result in significantly disproportionate/skewed training-testing subsets.

2.2 Decision Tree

The first step, after having investigated and preprocessed the data was to fine-tune the hyperparameters to (near) optimal values in order to be able to do the task at hand. The three main parameters that were focused on here were `max_depth`, `min_samples_splits`, and the `min_samples_leaf`, and these parameters are discussed in more detail in section 4.1.

The procedure followed for fine-tuning these hyperparameters is quite similar to that used for the subsequent model as well (logistic regression). Using a pipeline involving the `StandardScaler` from `sklearn.preprocessing` and a balanced decision tree classifier model, various combinations of hyperparameters were tested using the `GridSearch` module. This module used 3-fold cross

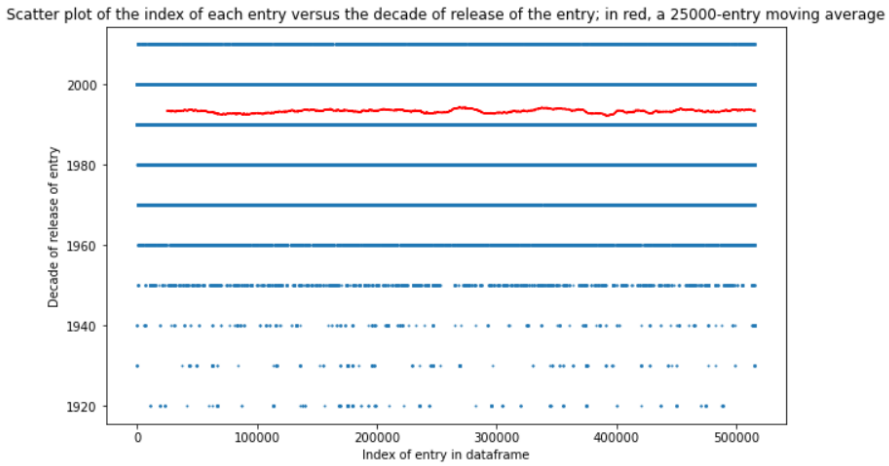


Figure 2: A scatter plot of the decade of each entry against it's index, along with a 25000-entry moving average.

validation, and trained 3 models for each combination of the 576 parameters and used the `score` method of the decision tree classifier as the evaluation metric to judge which combination performed best.

Given the size of the dataset being dealt with and the number of combinations the `GridSearch` module needed to try, the decision was made to run the search on a sample dataset 40% in size of the original dataset, consisting of entries chosen at random in order to save time, as well as computational resources.

2.3 Logistic Regression

Similarly, the hyperparameters needed to be fine-tuned for this model. Since the data had already been reduced down into the 12 principal components and the `lbfgs` solver only works with an `l2` penalty, the only hyperparameters to be tested were `max_iter` and `C`. The details of these hyperparameters and their influence on the model are discussed in section 4.2.

Following the same method as used for the decision tree classifier model, a `GridSearch` was used along with a pipeline involving the `StandardScaler` from `sklearn.preprocessing` and a balanced logistic regression model. Furthermore, the `GridSearch` module used 3-fold cross validation, and trained 3 models for each combination of parameters (70 in total) and used the `score` method of the logistic regression classifier as the evaluation metric to judge which combination performed best.

2.4 Evaluation

The function to evaluate the results of each classifier is defined in `full_clf_report`. This function displays a classification report and ROC-AUC score, both found in the `sklearn.metrics` library. The report also includes graphical representations of classifier performance, displaying a multi-class confusion matrix and multi-class ROC curve. Since there were no singular functions in sci-kit learn for displaying multi-class ROC curves and confusion matrices, those functions were implemented in `show_confusion_matrix` and `compare_multiclass_roc_curves`. The results of both classifiers are displayed in section 3, and discussed in section 4.

3 Results

3.1 Decision Tree

The values found for the optimal parameters for the decision tree model are as follows:

```
DecisionTreeClassifier(max_depth=35, min_samples_split=5, min_samples_leaf=1, class_weight='balanced')
```

Note: From the `GridSearch` module, the optimal value found for the arrtibute `min_samples_split` was 10, because that was the lowest number available in the testing range provided. However, during independent trials afterwards, it was noticed that values lower than 10 provide higher accuracy in the testing data as well as having a larger AUC.

3.2 Logistic Regression

The values found for the optimal parameters for the logistic regression model are as follows:

```
LogisticRegression(C=166.81005372000558, class_weight='balanced', max_iter=250
```

Useful in the evaluation of the model, the classification report, confusion matrix, and multi-class ROC curves are also shown. These results are discussed in section 4.2.

3.3 More Results

```
training accuracy: 0.9497250065888363
testing accuracy: 0.4673946279047888
Classification Report:
              precision    recall  f1-score   support

        1920       0.11      0.14      0.13        69
        1930       0.08      0.14      0.10        76
        1940       0.03      0.05      0.03       120
        1950       0.08      0.13      0.10      1030
        1960       0.10      0.12      0.11      3934
        1970       0.14      0.17      0.15      8107
        1980       0.19      0.23      0.21     13948
        1990       0.31      0.33      0.32     41118
        2000       0.68      0.62      0.65     98541
        2010       0.04      0.06      0.05      3121

    accuracy                           0.47    170064
   macro avg       0.18      0.20      0.18    170064
weighted avg       0.50      0.47      0.48    170064

AUC score:  0.5582815233745488
Confusion matrix, without normalization
```

Figure 3: Classification Report of the Decision Tree Classifier Model

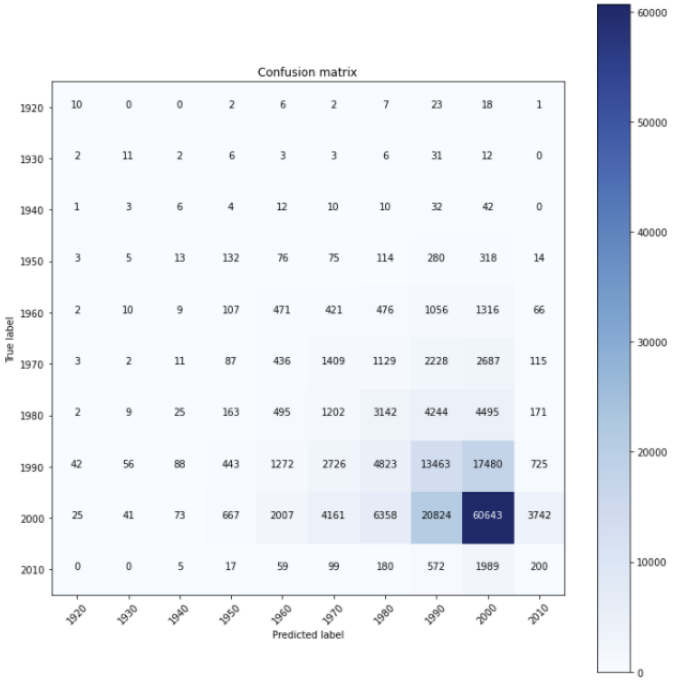


Figure 4: Multi-class Confusion Matrix of the Decision Tree Classifier Model

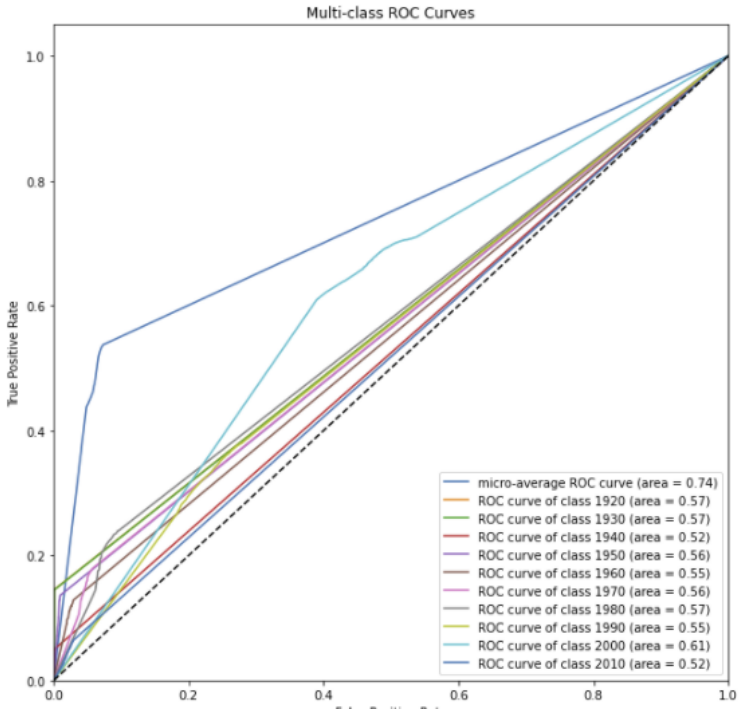


Figure 5: Multi-class ROC Curves of the Decision Tree Classifier Model

```
training accuracy: 0.23597591526901276
testing accuracy: 0.23562893969329193
Classification Report:
              precision    recall  f1-score   support

        1920       0.03      0.67      0.05        85
        1930       0.02      0.53      0.04        76
        1940       0.01      0.51      0.03       118
        1950       0.04      0.41      0.07      1031
        1960       0.09      0.30      0.14      3828
        1970       0.17      0.38      0.23      8202
        1980       0.21      0.47      0.29     13950
        1990       0.35      0.19      0.24     41088
        2000       0.75      0.20      0.31     98607
        2010       0.03      0.45      0.06      3079

    accuracy                           0.24    170064
   macro avg       0.17      0.41      0.15    170064
weighted avg       0.55      0.24      0.28    170064

AUC score:  0.6615838599365376
Confusion matrix, without normalization
```

Figure 6: Classification Report of the Logistic Regression Model

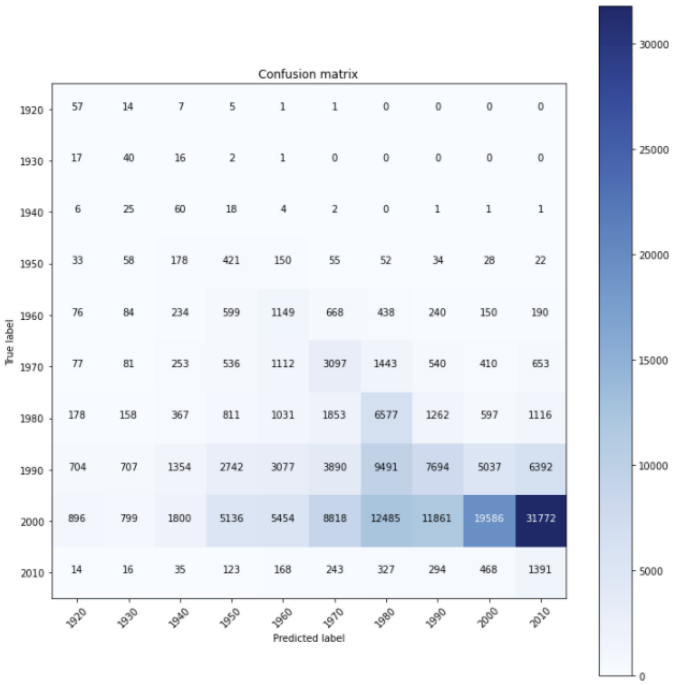


Figure 7: Multi-class Confusion Matrix of the Logistic Regression Model

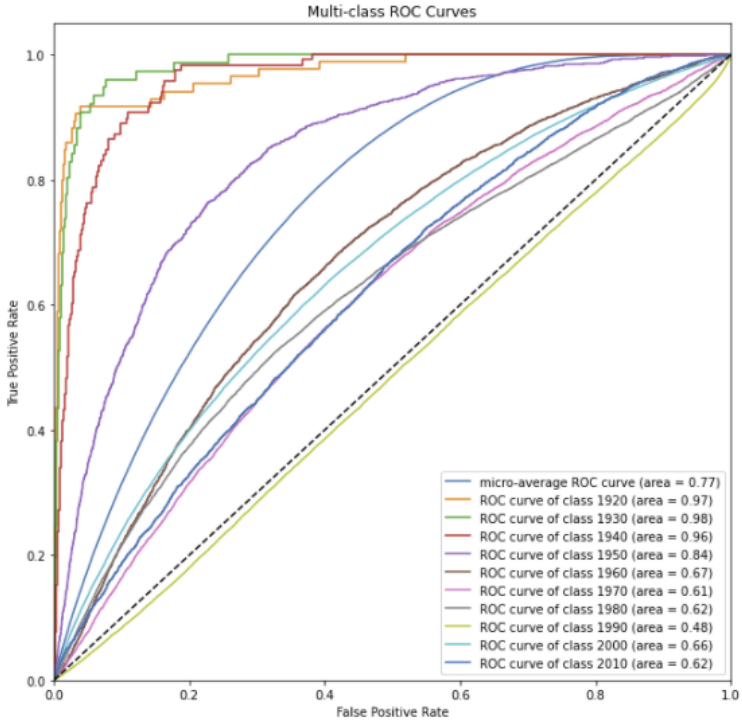


Figure 8: Multi-class ROC Curves of the Logistic Regression Model

4 Discussion

4.1 Decision Tree

Hyperparameter Results

A number of parameters were fine-tuned, starting with `max_depth`, and this is the maximum depth to which a tree model will be constructed. This needs to be tuned to optimality because if the value is too high, the tree model will be very complex since there will be more splits. This leaves room for overfitting the data if not paid attention too. However, having a low value may underfit the model and lose accuracy and information.

Next, the parameter `min_samples_splits` was tuned, and this is important for a similar reason of avoiding overfitting. If the values are too high, they may prevent the model from learning relations in the data which may be extremely specific to the training data, whilst making the values too high may result in underfitting [1]. Both extremes should be avoided.

Lastly, `min_samples_leaf` dictates the minimum number of samples required in order for a node to be a leaf node, and this is meant to ensure that the tree model does not overfit the training data by forming small branches to separately cater for certain parts of the data.

The last two parameters have been mentioned to be amongst the most important when it comes to determining the performance of a decision tree model [1].

Classifier Evaluation

Analysing the output of the classification report for the decision tree model as seen in fig. 3 elaborates on the scores achieved by each class through the prediction process, and we can see that the accuracy on the training data is far higher than that on the testing data. This may be a sign of some form of overfitting in play wherein the trained model did not generalise well to unseen data; the training accuracy hovers around 95% and that of the testing data is around 47%.

The values seen for the precision, recall, and f1-score appear to increase with the decades (with the exception of the 2010s because of how much less data was available in that class). This is mostly seen with the 1990s and 200s, and might mostly be a result of the fact that data from those two classes made up an extremely high proportion of the original dataset.

Moving on to the ROC seen in fig. 5, all classes performed quite similar, with the 1920s and 2000s being the closest (and the two highest performers). It is also noteworthy that the performance of the model was better than random guessing in all cases, however, not by much — with a lot of classes having an AUC just marginally higher than 0.5.

Lastly, as for the confusion matrix as seen in fig. 4, we can see that the most True Positives were seen in the class 2000s, most likely for the same reason of constituting a large amount of the dataset. It may also of be of interest to investigate the large number of misclassified entries in the classes 1990, 2000, 2010; potentially indicating an overall convergence in the style of music released in those eras.

4.2 Logistic Regression

Hyperparameter Results

The number of maximum iterations is important to optimize since too many iterations would result in overfitting the dataset as well as unnecessary complexity. Too few iterations could result in a model which has not learned all the useful information about the dataset it could. The second hyperparameter, C, represents the inverse of regularization strength. Higher values of C correspond to less regularization of the data, meaning the model may tend to overfit the training

data, causing larger discrepancies between training and testing accuracy and less generalizability. The optimal combination for these hyperparameters found by the `GridSearch` module is `max_iter=250, C=166.81`. These values point to a low amount of regularization for the model, indicating that it may be overfitting on the training data. However, the data is complex enough where this is understandable, and it is unlikely that there is one generalizable model which can represent the decade splits. Logistic regression is only optimal for linearly-separable problems, and due to the amount of complexity and variance in the music data, it would be naive to assume that logistic regression would be able to have strong regularization and expect accurate results.

Classifier Evaluation

The classification report shown in fig. 6 distinguishes between the scores of each decade, making it possible to see the subtle differences in the success of the classifier for each decade. In total, the accuracy of the classifier is around 24%, and there was no significant difference between the training and testing accuracy, indicating the model did not drastically overfit the training data.

The values for precision, recall, and the f1-score are much higher for the decades which had the most support in the dataset, namely, the 2000s and 1990s, with the other decades having significantly lower scores in all three evaluation categories. However, the earlier decades had much better AUC scores, demonstrated by the ROC curves shown in fig. 8. This would indicate that the classifier had an easier time classifying music from earlier decades, possibly because the songs from those eras are fairly distinct in musical terms.

All of the decades perform better than random guessing with the exception of the 1990s, which performs just worst than random guessing, as shown in fig. 8. Analysis of the confusion matrix in fig. 7 shows that this is because a majority (9,491) of 1990s songs were misclassified as being from the 1980s. This may indicate that the classifier did not develop a suffcient model to distinguish the unique qualities of music from the 1990s from 1980s music.

The testing data from the 2000s made up a majority of the testing data and its row in the confusion matrix indicates some interesting features of the model. Most glaringly, it appears that a majority of 2000s music was misclassified as being from the 2010s. This may be because the sample size for 2010s music was incredibly small, with the most recent song being from 2011, and it is unlikely that the classifier had enough information to be able to truly distinguish the decades apart musically. Additionally, since the classifier was trained on a balanced dataset, the weight of music from the 2010s equaled music from the 2000s, despite there certainly being a much broader range of style and sound in the data from the 2000s than in the 2010s. The second most-common misclassification of 2000s music was into the 1980s category. Which could indicate that 2000s music has more timbre similarities to 1980s music than 1990s music, but more likely indicates a deeper problem about the classifier's ability to distinguish 1990s music from its surrounding decades.

Overall observations about the confusion matrix and ROC curves would indicate that music of the 1980s and beyond are much more similar stylistically than any of the previous decades, and distinction between decades begins to blur as the quantity and stylistic range of the music grew.

4.3 Classifier Comparisons

Some similarities in the performance of the two models:

- They both tend to classify songs better than random chance in nearly all classes
- They both indicate greater stylistic similarities in the music released 1980 onward.
- Both have better results for precision, recall, and f1-score for the classes 1980, 1990, and 2000, presumably because of how significant a portion of the dataset these classes make up.

Some of the differences between their performances:

- The overall accuracy scores of the Decision Tree Classifier are noticeably higher than that of the Logisitic Regressor, however this may be as a result of the Logistic Regressor being better suited for linearly separable data.
- The Logistic Regressor outperforms the Decision Tree Classifier when it comes to using the ROC as a metric of comparison, with the Decision tree barely performing better than random chance for nearly all classes.

5 Limitations

When working with these classifiers and more specifically this dataset, it is beneficial to be aware of the reasons for limitations in the analysis being done:

- Genre differences within the songs were not taken into account — there are going to be songs from the same era that belong to different genres, and therefore may sound unlike one another.
- The decision tree model had it's hyperparameters tuned to a relatively small portion of the complete dataset to cope with extensive computational load of that process, and this implies some room for increased precision in the values picked.
- Logistic regression classifiers tend to only work on linearly-separable data and is better at generating simpler models. Due to the extreme nuance involved in identifying a song's decade based on its timbre, using a more complex classifier such as a neural network may produce more accurate results.

6 Recommendations and Further Research

Further research could be done into exploring more specific era definitions, not just ones defined by 10-year time-periods. It may be possible to use unsupervised clustering or some similar technique to identify inherent shifts in musical style through non-constant time-periods, and more intentionally define these musical "eras". This would most likely give more accurate results, since the model would have more cohesive stylistic categories to identify, rather than classifying all music released between 2000 and 2009 as "2000s music".

As it stands, there are a great number of applications for a model which can classify a song into a certain decade based only on its musical characteristics. Since the phenomenon of recorded music is quite recent, the release date of a song is usually well-documented, and such a model would be more useful in identifying what a given song *sounds* like it's from. This could be used in making decade-themed playlists, or identifying historical influence in the music of modern artists. From this use, it's actually beneficial that the model doesn't classify all songs with perfect accuracy, as long as the misclassifications do indicate some meaningful stylistic influence.

7 Appendix

The Jupyter notebook containing the code: Github Repository, Music Year Prediction.

References

[1] Rafael Gomes Mantovani, Tomás Horváth, Ricardo Cerri, Sylvio Barbon Junior, Joaquin Vanschoren, and André Carlos Ponce de Leon Ferreira de Carvalho. An empirical study on hyperparameter tuning of decision trees. *CoRR*, abs/1812.02207, 2018.