Music Tribe Report - Joshua Uwaifo

Joshua Uwaifo
(MSc) University of Edinburgh, (BSc) University of Warwick

1 Introduction

1.1 Problem setting

Music Information Retrieval (MIR) is the art of retrieving insight, such as, recommending music, finding the title of a piece of music or extracting the mood that a piece of music conveys. In this project provided by Music Tribe, the task is to classify a given set of audio samples into 9 instrument categories (including vocals). Music Tribe is a company that has provided and continues to provide state-of-the-art and hopefully next generation audio tools using MIR.

1.2 Data set

The data set provided by Music Tribe has a data folder consisting of 9 subfolders which serve as the categories or labels. These are namely; bass, guitar, hi-hat, kick, piano, saxophone, snare, tom and vocals. Each sub-folder consists of json files. These datafiles consists of metadata like frame size, hop size, the number of samples and sample rate alongside the data features.

Before delving into the data features, it was important to see if the meta data features are fixed across all json files over the 9 categories. This is because consistent comparisons need to be ensured otherwise further pre-processing needs to take place. This preliminary investigation is detailed in the ipython notebook titled "Initial investigation". From the investigation, it was clear that the meta data was consistent apart from the variation in the number of samples across different datafiles. The investigation also showed that the number of json files is not the same as the number of samples and also hinted at the possibility of class imbalance across the 9 categories. The class imbalance was hinted because all the category subfolders had 400 files apart from the piano and saxophone categories which had 8 and 23 files respectively. This possibility of class imbalance also inspired the idea of using a confusion matrix alongside the general accuracy when evaluating the models chosen.

2 Exploratory data analysis

After loading using the utility in the python file named "util", the cleaned up dataset consisted of 8,869 samples and 46 features. These features include 13 coefficients for GFCC (gammatone frequency cepestral coefficient) and MFCC (mel-frequency cepestral coefficient) alongside 11 coefficients for LPC (linear predictive code). The remaining 9 features are namely; centroid, end, energy, flatness, flux, spectral complexity, start, zero cross rate and the onset time.

As alluded to previously, but confirmed in doing Task 2, class imbalance exists in the dataset provided.

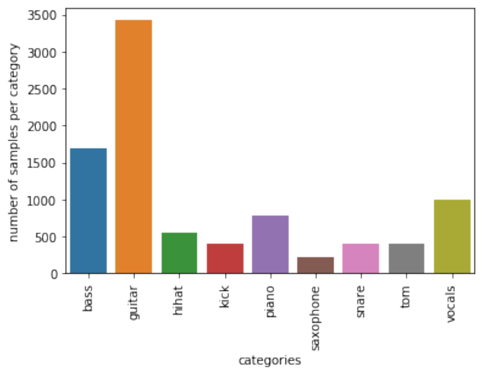

Figure 1: Class Imbalance

Here we see that, the most frequent class is the guitar category which is double that of the second most frequent class, bass. The least frequent category was the saxophone class. Figure 1 shows a clear variation in the number of samples category, further emphasizing the need for the evaluation method to takes this into account.

In addition, the exploratory analysis showed that no features consisted of constant values. If constant feature values did exist, it would suggest redundant data is present and would have needed to be cleaned up as an extra pre-processing step.

Furthermore, it was interesting to see the pair-wise correlation between the 46 features. A further utility was created that obtained the top 10 and lowest 10 absolute correlations. The most highly correlated attributes were the start and end features, which makes sense as the data provided was probably segmented in a fixed frame. These features had an absolute correlation of 0.9998 (4s.f), with perfect correlation having a value of 1. On the other hand, the lowest 10 correlations showed that quite a few features have little to no pairwise relationship with the lowest 10 all having values around 0. The pair of features with the least correlation were the centroid and end features. The exploratory data analysis can be seen further in the ipython notebook titled “Tasks 2 and 3”.

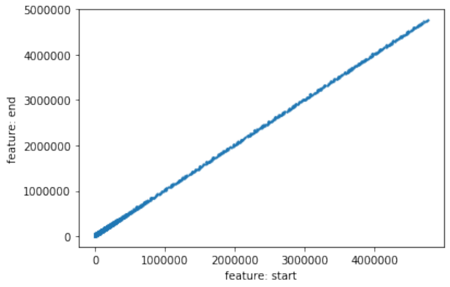

Figure 2: Highest correlated features

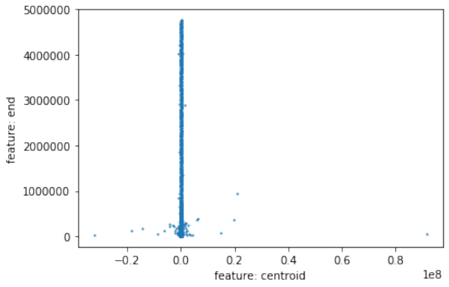

Figure 3: Lowest correlated features

Finally, the mean of the 46 feature vectors were investigated. Statistics such as the standard deviation and mean of the averages show a wide spread across the feature values which means comparing them without normalising could pose a problem. As a result, the dataset was chosen to be standardized as well as keeping a copy of the original dataset. Standardization is a normalization technique that ensures each feature in the training set has a mean of 0 and variance or standard deviation of 1. This can and should enable better performance on certain classifiers, that are affected by extreme numerical values, seen especially in outliers. The statistics around the mean of the feature vectors are seen in the ipython notebook mentioned previously.

3 Classification models

In order to evaluate the classification models, I first of all partitioned the dataset into training, validation and testing dataset splits. The splits were in a 60:20:20 ratio, ensuring enough samples in the training (around 5,300) alongside large enough evaluation datasets (1,774 samples). This technique is called double hold-out validation. The reason for the validation dataset here is to decide between model choices and the testing dataset is for reporting a score for how well the final chosen model generalizes to new data. Also, due to the class imbalance mentioned previously, a confusion matrix is used to see how well the model performs relative to the truth in predicting each of the 9 categories. The accuracy of the models are also included for a more concrete evaluation score.

Having defined the relevant evaluation metrics, I finally attacked the problem of identifying the instrument type from a given audio sample. The main multi-class classification techniques that were used were the K-nearest neighbours classifier and a decision tree. In addition, two baselines were investigated, a majority and a stratified class classifier. As an aside, the use of a fixed random seed ensures the models and data splits are reproducible, in case the reader wants to verify.

As alluded to earlier, each of these techniques are also evaluated on the original and also standard scaled version of the datasets.

Firstly, the majority class (most frequent) classifier was a better baseline than the stratified baseline in terms of accuracy. However, the fixed prediction makes the predictions less varied than the stratified model in terms of confusion matrix performance. Here, standardising the dataset does not affect the performance of the baseline models used. To summarise, the majority class classifier had an accuracy of 38.6%.

Regarding the non-baseline models, the two model types focused here were an instance-based and non-instance based model. The instance based model of choice was the K nearest neighbour (KNN) classifier. The non-instance based model of choice is a (piece-wise) linear model, the decision tree classifier.

3.1 Top performing model: 1 Nearest Neighbour, standardised dataset

The model type that performed best on the validation dataset was the 1-Nearest neighbour, specifically on the standardised dataset. This model had a validation accuracy of 95.49% (4 s.f). For comparison, the model with the second best validation accuracy was the Decision Tree model with an accuracy of 91.15%. As a brief aside, regarding the decision tree, standardising the dataset made no difference due to the piece-wise linear nature of the model. The analysis of the top two models confusion matrices (Figure 4 and 5) emphasise the superior performance of the 1-Nearest neighbour model.

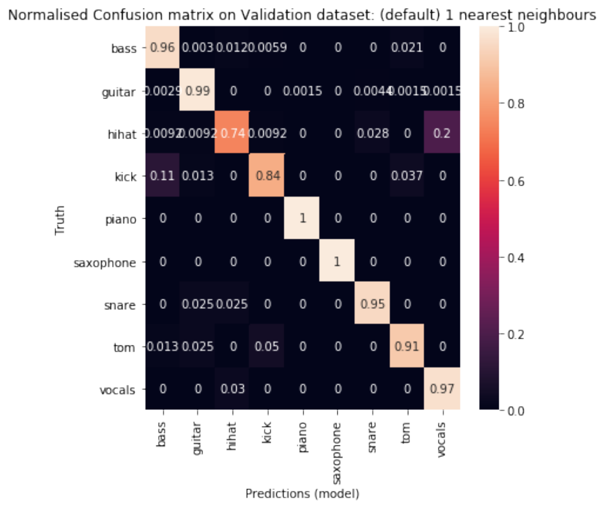

Figure 4: 1 Nearest Neighbour validation performance

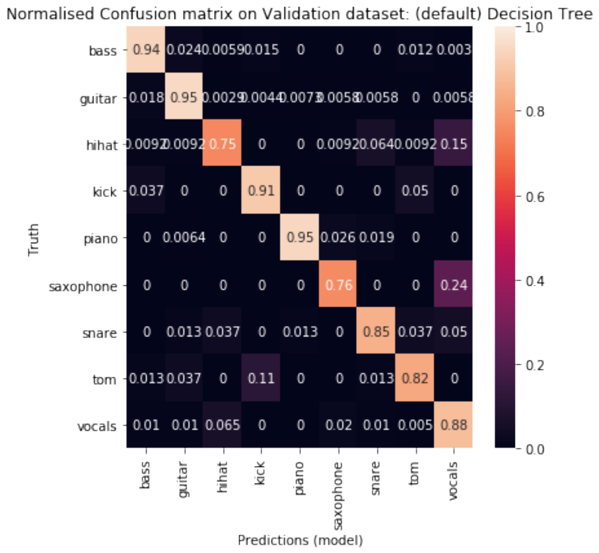

Figure 5: Decision Tree validation performance

In comparison to the decision tree model, the 1 nearest neighbour classifier on the standardised dataset performed better in predicting every category except the hi-hat. The nearest neighbour classifier was perfect in predicting a piano and saxophone. It did perform worse by 1% to the decision tree when predicting hi-hat as it similarly, to the decision tree, had issues in distinguishing between the hi-hat and vocals. On the other hand, whilst the decision tree found it difficult in accurately predicting the snare, the nearest neighbour had no such problem. Generally, figure 4, shows that this model performs really well in the validation performance, making it the model of choice. It is also important to note that there is a huge decrease of almost 40% when performing K-nearest neighbour on an non-standardised dataset (accuracy of 56.82%). This further proves the impact of normalising the features for this model on this task.

3.2 Limitations and strengths of top performing model

Although the (1) nearest neighbour classifier performs well in this task, theoretically KNN is over-sensitive to outliers meaning extra care needs to be taken to ensure the data does not consists of outliers in practice. Furthermore, it is a computationally expensive training procedure taking O(nd) to compute the distances in training, with n being the number of samples and d the number of features. However, as KNN is an instance-based learner, if the training is done off-line then it is a one-off cost. In addition, this model makes no further assumption on the data except the implied distance function (euclidean being the default for sci-kit learn). Finally, effective design choices can be made to implement more efficient representations of the data and with the model only requiring the nearest neighbour, this is really promising and should be doable in practice.

4 Conclusion

The (1) nearest neighbour classifier on the standardised dataset shows really promising results off-line in categorising the instrument of a given audio sample. In order to scale this up for use in a product, we would need to ensure a quick and effective translation between the live audio stream and the audio pre-processing that produces the samples in the format provided in this dataset. In addition, the standard scaling object that normalised the training dataset would also need to be used to transform new audio samples. Training off-line might also need to be done, however, due to only the nearest neighbour being needed, there is a possibility of performing this on-line too. To conclude, the generalisation performance of the 1-nearest neighbour on (unseen) testing data is 94.87% (4s.f) with the following confusion matrix. Thank you so much for this project, it was fun!!! I hope I answered your questions effectively, sorry for the extreme detail looool.

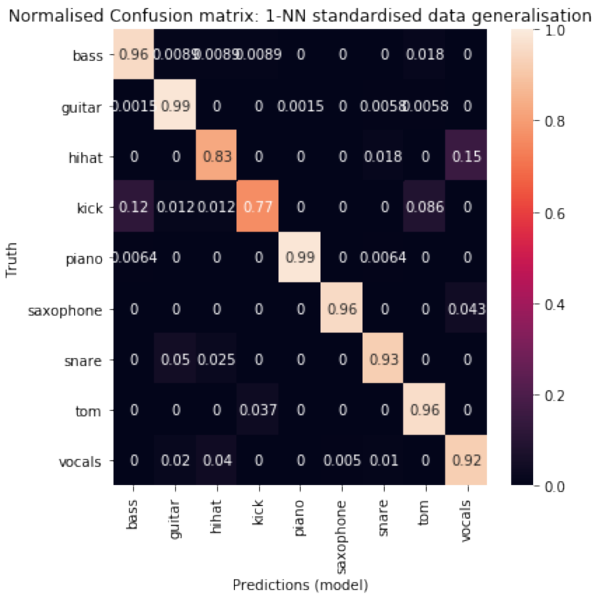

Figure 6: 1 Nearest Neighbour generalisation performance