A machine learning approach to automatic music genre classification: comparing different methods

***Abstract*—As machine learning starts to take over the world, it has become evident to scientists and companies the broad amount of general problems with daily effects that these methods can solve. Even more so, these techniques have proven useful in a wide repertoire of areas even in the world of music. The main issue surrounding this work is that of classifying a song into a specific genre. The identification is carried on through the most evident feature, that is, a song is classified into genre X if its characteristics point more to one genre even if many others are present. In fact, we propose several possible approaches to automatic music genre classification that have shown promising results. The several methods explored are explained thoroughly and are thus compared using two main metrics: accuracy and log loss. The findings of this work point towards gradient boosting as the most effective and far reaching method to tackle the issue at hand, achieving the most solid metric results amongst all the approaches.**

Keywords*—*Classification, Gradient Boosting, Logistic Regression, Machine Learning, Music Genre, Support Vector Machines.

I. Introduction

With the advent of services dedicated to spreading, distributing and commercialising music, music has become a signicant part of the Internet content. Due to this fact, Music Information Retrieval (MIR) has become an important research area, focusing on automatic procedures capable of dealing with large amounts of music in digital formats.

The task of Automatic Music Genre Classication (AMGC) is one of the problems studied by MIR. Musical genres are categorical labels that are used to characterize music, created by human experts in order to identify the style of the music, and are largely used to organize collections of music in digital formats [1].

Although there is no industrial standard [2] and the division of music into genres can be somewhat subjective and even arbitrary, there are criteria related to the timbre, pitch — i.e., melody and harmony — and rhythmic structure of music that can be used to characterize a particular genre.

The AMGC problem was initially formulated in the work of Tzanetakis and Cook [3], in which a comprehensive set of features was proposed to represent a music piece, including timbral texture features, beat-related features and pitch-related features extracted from 1000 samples from 10 different music genres. Employing Gaussian classiers, Gaussian mixture models and the k Nearest-Neighbors (k-NN) classier, they obtained results which indicate an accuracy of about 60%, using a ten-fold cross validation procedure.

In this paper we analyse several machine learning approaches to the issue of AMGC. The task of AMGC is a difficult multi-class classification problem: while differentiating some genres is fairly straightforward, the distinction between others is more subtle.

One possible solution is to decompose the original multi-class problem space into a series of binary classication problems, according to a One-Against-All (OAA) approach [4]: a classier is constructed for each class, and all the examples in the remaining classes are considered as negative examples of that class; new data should then be classified to the class which classifier yields the highest probability of said data point belonging to that class.

This document is organised as follows: section II describes the employed datasets; III ; section IV presents the experimental results; finally, section V summarises the conclusions of our work and identifies possibilities of future work.

II. Data analysis

In this work, we used two different datasets: a training dataset with 4363 entries, and a test dataset with 6544 entries. These are a custom subset of the Million Song Dataset.

Each entry corresponds to a representation of a song in 264 distinct features which are a summary representation of the 3 main components of music: timbre, pitch (melody and harmony) and rhythm. For simplicity, each song has been assigned only one label — obtained from AllMusic.com — corresponding to the most representative genre among the 10 possible genres considered:

1) Pop-Rock
2) Electronic
3) Rap
4) Jazz
5) Latin
6) RnB
7) International
8) Country
9) Reggae
10) Blues

By looking at Fig.1 and analysing the class distribution within the training dataset, it is plausible to expect that the uneven class distribution within the training dataset may be an issue. In fact, classification algorithms are usually biased towards the majority class and can show deceivingly high overall prediction accuracy, while at the same time exhibiting poor performance in the prediction of other minority classes. As such, the imbalance in class distribution within the training dataset and particularly the obvious disparity between the number of songs classified as belonging to the Pop Rock genre and the number of songs belonging to other genres within the training dataset may lead to the creation of biased prediction

models, particularly if the genre distribution within the test dataset does not follow this pattern. Altering class distribution so as to produce a more balanced class distribution can be an effective method for alleviating the adverse impact of class imbalance.

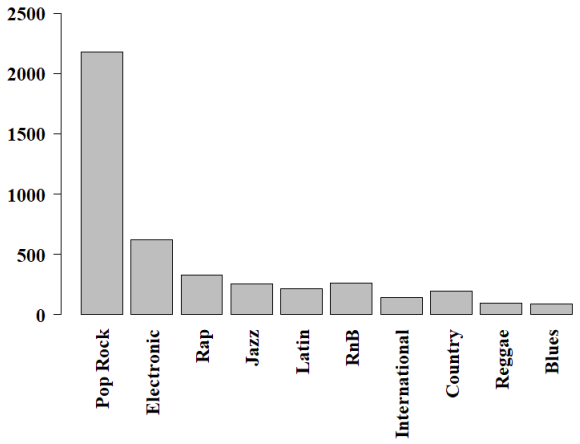


Fig. 1. Number of songs for each genre, representing the genre distribution on the training dataset.

Furthermore, given the dimensionality of the feature vectors — 264 distinct features —, some redundancy in the data is to be expected. In fact, Fig. 2 suggests that several features are indeed redundant, as they display a strong degree of correlation. This high dimensionality will increase computational complexity, increase the risk of over-fitting and the sparsity of the data will grow. Hence, dimensionality reduction would limit these phenomena.

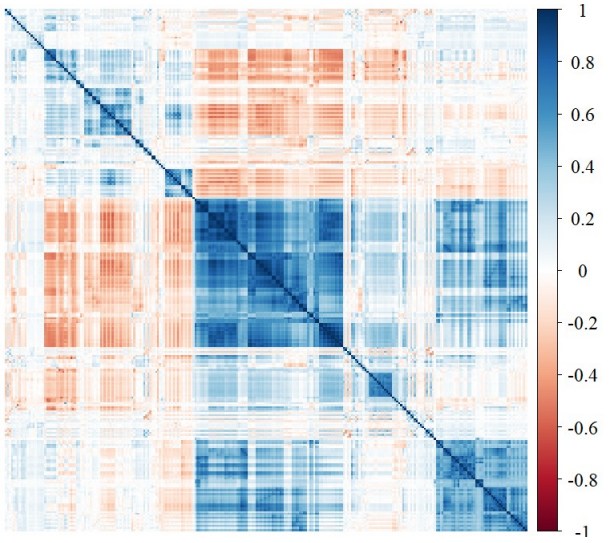


Fig. 2. Correlation matrix between the features for the entries on the training dataset.

In the figures on Appendix A, we visualise the data within the training dataset, by plotting the range of the values of each feature, for each particular genre.

III. Methods and Experiments

Our approach involved experimenting with different standard algorithms for supervised machine learning to accomplish the AMGC task.

Simply put, we used three different classification techniques: Logistic Regression, Support Vector Machines and Gradient Boosting. Several experiments were conducted using each of these methods:

- Logistic Regression
- Logistic Regression with feature scaling
- Support Vector Machines
- Support Vector Machines with Grid Search for parameter optimisation
- Gradient Boosting

A. Logistic Regression

Logistic regression is a particular case of regression that makes use of the sigmoid or logistic function $\sigma(z)$ in which z is the dot product of a weight vector and the feature vector $(w^T x)$. This model has a probabilistic interpretation which represents the conditional probability of a label being classified 1 knowing x as $h^{(w)}(x)$ and the probability of being classified 0 knowing x as $1-h^{(w)}(x)$. This means that the solution space is divided into two areas of classification with a sigmoid curve as a decision boundary.

The first method implemented was logistic regression based on the OAA approach. Several classifiers were employed - one for each genre - in which each would ascertain if the song belongs to its genre or not, with the final genre classification being selected based on the *a posteriori* probability of each classier. This was implemented through the use of the library **sklearn.linear_model**[1].

We repeated the experiment, performing feature scaling on both the training and test datasets, so as to ascertain whether this had any staggering impact on the result.

B. Support Vector Machines

Again, a set of possibly different candidate classes is produced by the individual classiers, composed to produce the nal genre label, following the same OAA approach described before.

This time, we employed Support Vector Machine (SVM), a supervised classification method that finds the maximum margin hyperplane separating two classes of data. The space of possible classifier functions consists of weighted linear combinations of key training instances in this kernel space. The SVM training algorithm chooses these instances (the support vectors) and weights to optimise the margin between classifier boundary and training examples. Since this classification method usually performs well in high dimensional spaces, its use aligns nicely with the problem of song genre classification.

[1]Documentation can be found here: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

The implementation itself was achieved by using the library **sklearn.svm**[2]. As this algorithm is not scale invariant, feature scaling was applied to both the training and test datasets. In this first attempt, no parameter tuning was performed.

Next, we experimented with parameter tuning by using **sklearn.model_selection.GridSearchCV**[3], as illustrated on Appendix B This method performs an exhaustive search over the specified parameter values for the model provided — in this case, SVM. The parameters of the model are optimised by cross-validated grid-search over a parameter grid.

C. Gradient Boosting

Boosting methods saw an increase in popularity in the close past. Being methods that rely on the principle that a weak learner can be improved, the idea behind them is filtering observations in which the method miss-classified and get a succession of hypothesis from them.

Gradient boosting creates a single predictive model consisting of a series of simple decision trees. It is an ensemble algorithm that aims to improve accuracy by minimising the error term [5]. Following the momentum from the previous constructed models, the **sklearn** library was employed, more particularly the **sklearn.essemble**[4]. The parameters fed to the model were the following:

```
loss = 'deviance',
random_state = 10,
performCV = True,
printFeatureImportance = True,
cv_folds = 5
```

code/gb_parameters.py

This means that the loss function to be optimized was the so called 'deviation' and the parameterisation of the $random\_state$ which attends to the random seed generator[5] was set to 10.

Since this model is subject to many hyper parameters, some variations were introduced in some of the variables. For example, for a learning rate the values 0.1, 0.01 and 0.001 were tested to which it was observed that the value 0.1 performed the best. Further experiments were carried out in respect to $warm\_start$[6] either tuning it to true or false. Moreover, some variations were introduced on the max_depth[7]. Variations were in between the values 3, 5 and 10. The best result was found with $max\_depth = 5$.

[2]Documentation can be found here: http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[3]Documentation can be found here: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[4]Documentation found here http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.GradientBoostingClassifier

[5]From the documentation "If int, $random\_state$ is the seed used by the random number generator"

[6]Documentation on $warm\_start$: "When set to True, reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just erase the previous solution".

[7]Documentation on max_depth: "maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree. Tune this parameter for best performance; the best value depends on the interaction of the input variables".

The results presented correspond to those correlated to the best obtained with parameters out of those listed. So, ultimately, these featured accuracy and log loss metrics were obtained through the use of the following settings:

```
learning_rate = 0.1,
warm_start = False,
max_depth = 5,
random_state = 10
```

code/gb_hyper.py

IV. Results

Two main metrics were employed in order to evaluate the performance of the methods that were implemented.

a) Accuracy: The first one, **accuracy**, reflects how close the classification is to the real genre obtained originally from AllMusic.com in a straightforward way. This means that the highest the value obtained the better. In fact, the metric will tell us a quantification of the number of times our machine learning algorithms are identifying the songs correctly. The results obtained for this metric are shown on Table I. The accuracy values were obtained on the **Kaggle** platform using their validation mechanism.

TABLE I. Results obtained on Kaggle using the accuracy metric.

Method	**Optimisation**	**Accuracy Score**
Logistic Regression	-	0.62454
Logistic Regression	Feature Scaling	0.61002
SVM	-	0.51161
SVM	Grid Search	0.59321
Gradient Boosting	-	0.63600

b) Log loss: The most distinctive feature of the log loss metric from the accuracy metric is that it does not evaluate simply the genre chosen by the classifier for each observation but the list of probabilities that an observation has of belonging to each possible genre. This means the output file for a log loss metric has one column for each genre with a probability value that expresses how likely it is that the song belongs to the specific genre.

Log loss quantifies the accuracy of a classifier by penalising incorrect classifications. Minimising the log loss is basically equivalent to maximising the accuracy of the classifier. This means that this metric works in an opposite manner than that of accuracy - the smallest the value obtained, the better is the classification method. The results obtained from **Kaggle** are showcased on Table II.

TABLE II. Results obtained on Kaggle using the log loss metric.

Method	Optimisation	Log loss Score
Logistic Regression	-	0.19155
Logistic Regression	Feature scaling	0.24258
Gradient Boosting	-	0.17990

Classification accuracy alone can be misleading, particularly in the case of multiclass problems or when in the

presence of an unequal number of observations in each class, as it is the case. To get some insight not only into the errors being made but more importantly the types of errors that are being made, we generated a confusion matrix to summarise the performance of the classification algorithm used in each experiment, using the predicted classifications for each entry in the training dataset.

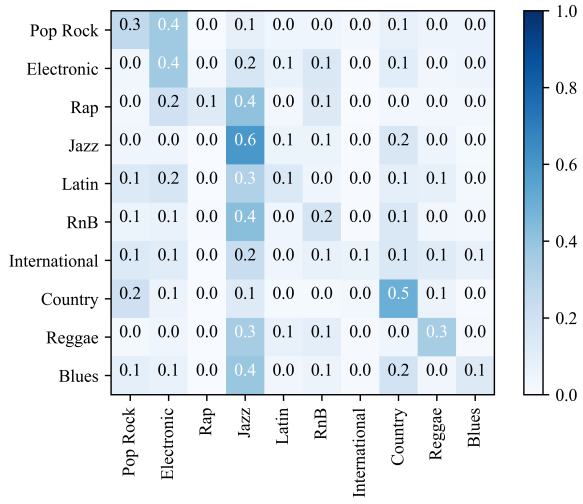


Fig. 3. Confusion matrix obtained for Logistic Regression.

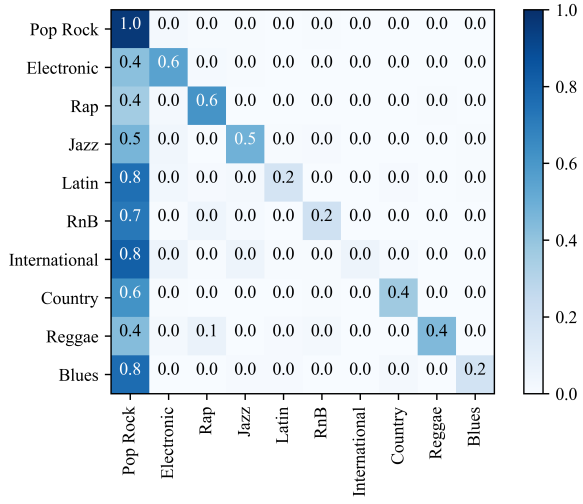


Fig. 4. Confusion matrix obtained for Logistic Regression with feature scaling.

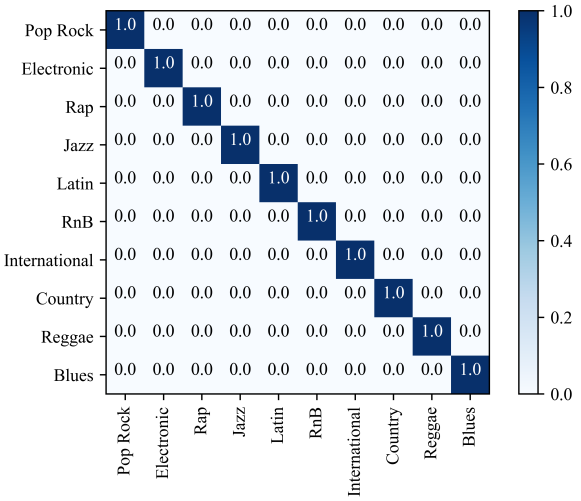


Fig. 5. Confusion matrix obtained for SVM.

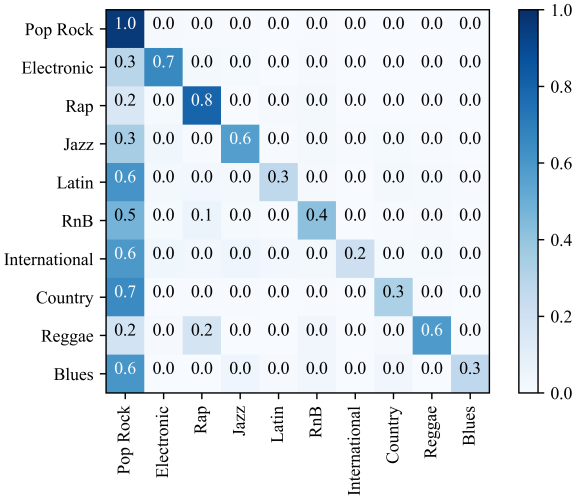


Fig. 6. Confusion matrix obtained for SVM with parameter optimisation.

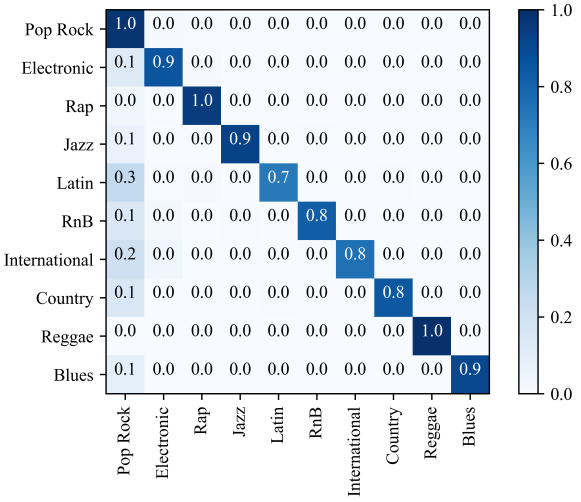


Fig. 7. Confusion matrix obtained for Gradient Boosting.

V. Conclusions

Several conclusions can be inferred from the results obtained.

The first is that, as the literature suggested, the use of a set of binary classiers is adequate in problems that present complex class separation surfaces, as it is the case with AMGC.

Looking at the results, some statements can be inferred. Our experiments showed Gradient Boosting to be the most effective approach for the problematic of AMGC, as it outperformed the other methods considered on both accuracy and log-loss metrics, closely followed by Logistic Regression.

An interesting observation was that of the finding that optimisation methods in Logistic Regression such as feature scaling did not improve the score attained by the regular approach. This leads to arising questions on the fit of this technique to the issue at study.

Again it is proven that the best approach to a machine learning issue is to work on an experimental basis, to try and to fail and to improve through the lessons learned.

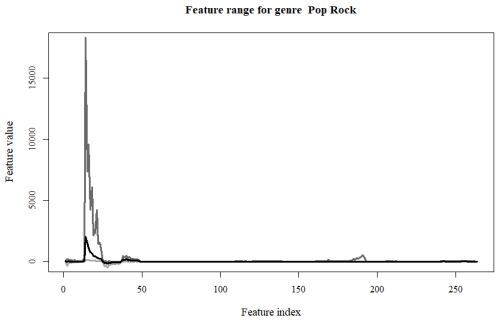
Even though SVM is an approach particularly well suited to handle high-dimensional data, only through the use of parameter optimisation were we able to increase classification accuracy significantly (Fig. 4). We can postulate that this is due to over-fitting, as by looking at Fig. 5 we can see that all entries on the training dataset were correctly classified, i.e. the accuracy for the training dataset is 100%, while the accuracy for the test dataset obtained from **Kaggle** is significantly lower.

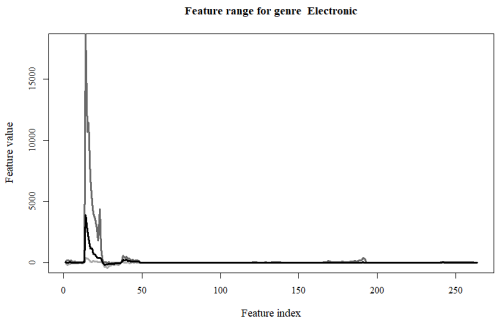
Gradient Boosting also displayed the least confusion in classification (Fig. 7). An overview of the confusion matrices generated reveals a tendency for other genres being erroneously classified as Pop Rock. This undoubtedly related to the imbalance in class distribution within the training dataset used.

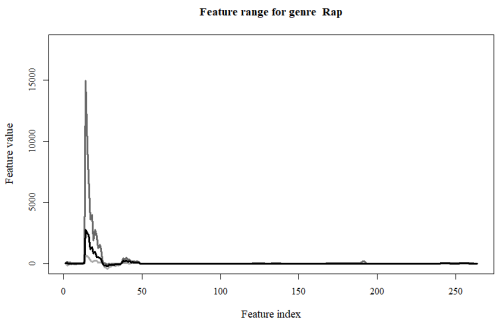
Both accuracy and the log loss metrics are sensitive to class imbalance. In this case, because our training dataset has a large number of entries with genre Pop Rock, a classifier that labelled all entries as Pop Rock would yield a high accuracy score and low log loss score, even though this classifier is clearly not satisfactory. In the case of log loss, the metric can be generalised to incorporate misprediction costs and be adjusted to account for class imbalance.

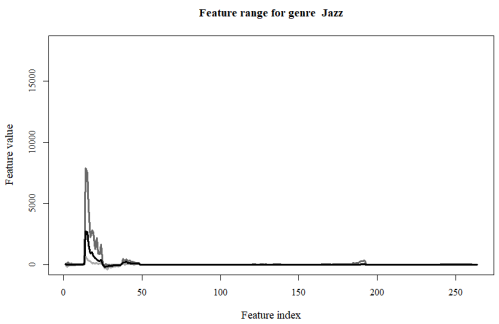
Leveraging on these results, further work could be conducted, be it analysing the performance of other approaches, such as Neural Networks, or by combining several models and averaging their results. This former approach, however, is a two-edged-sword, that is, the overall model might become too complex to be efficiently employed on a real life situation per say, on a daily used piece of software or in a company's data infrastructure. That is one of the core issues at the heart of machine learning — in theory one might improve models drastically but in reality this is not always the best answer to the problem at hand. In cases it might prove better to favour efficiency over accuracy at the cost of a not too impactful change in metric result.

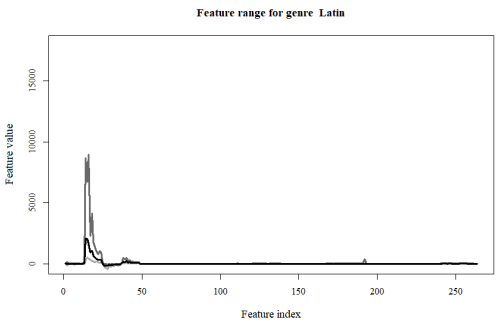
APPENDIX A
FEATURE RANGE WITHIN EACH GENRE

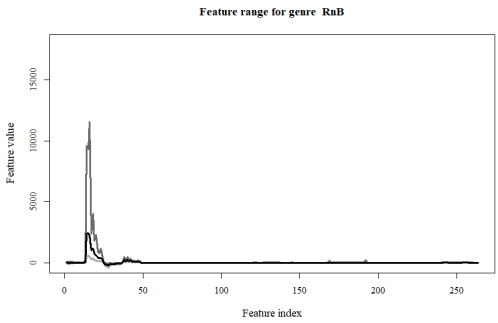


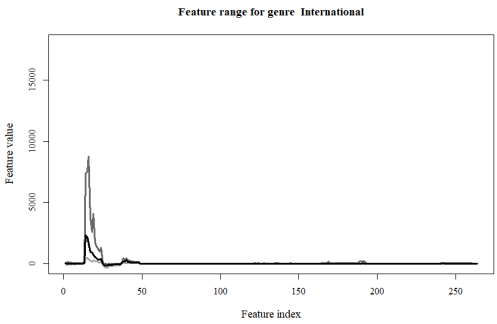


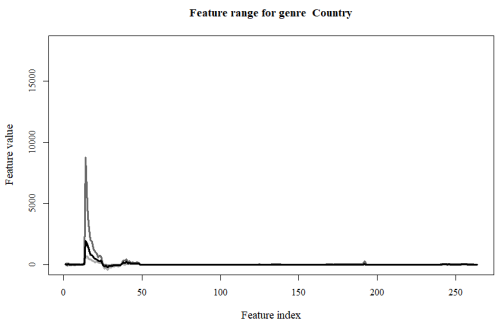


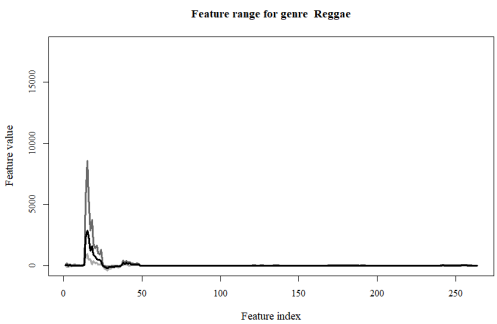


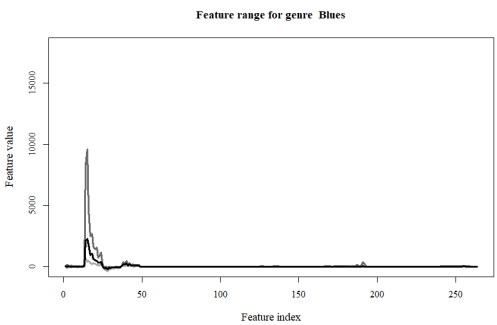












APPENDIX B
GRID-SEARCH PARAMETER OPTIMISATION

```
parameter_candidates = [
        {'C': numpy.logspace(2,8,6), 'gamma': numpy.
    logspace(-7,-1,6), 'kernel': ['rbf']}
    ]

# Create a classifier object with the
# classifier and parameter candidates
optimiser = GridSearchCV(
    estimator = svm.SVC(),
    param_grid=parameter_candidates)

# Train the classifier with train dataset
# and train labels
optimiser.fit(train_data, train_labels)
    classifier = svm.SVC(
    C=optimiser.best_estimator_.C,
    kernel=optimiser.best_estimator_.kernel,
    gamma=optimiser.best_estimator_.gamma)

classifier.fit(train_data, train_labels)
```

code/grid_search.py

REFERENCES

[1] Jean-julien Aucouturier and François Pachet. Representing Musical Genre : A State of the Art. *Journal of New Music Research*, 32(February 2015):83–93, 2003.

[2] Karin Kosina. Music genre recognition. *Master's thesis, FH Medientechnik und-design, Hagenberg, Austria*, page 84, 2002.

[3] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[4] Carlos N. Silla Jr., Alessandro L. Koerich, and Celso a. a. Kaestner. A machine learning approach to automatic music genre classification. *Journal of the Brazilian Computer Society*, 14(3):7–18, 2008.

[5] Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38(4):367–378, 2002.