

ABSTRACT

Today, the world is seeing an explosive growth of songs that are released every single day. Music streaming services like Anghami are facing the challenge of keeping up with classifying the genre of every newly released song. Hence, there is an increased interest in **automated** music genre classification which will help these services provide better music recommendation systems and allow users to identify music they like. This study aims at developing different machine learning models after applying pre-processing and feature engineering techniques, which can be used for automatic music genre classification. The goal is to try to improve the accuracy reached by other authors on the chosen dataset, which is the GTZAN dataset composed of 1000 audio songs. To try to reach an optimal accuracy, the dataset which is originally split into 30 second audio files was also split into 3 second files. An overall number of 83 features was extracted from each audio file using Python’s LibROSA audio analysis library. Seven classifiers including Gaussian Naïve Bayes, Stochastic Gradient Descent, K-Nearest Neighbors, Support Vector Machine, Logistic Regression, Random Forest, and Neural Networks (NN) were developed after splitting the data into training, validation, and test sets, and performing cross-validation. Then the models were evaluated and compared. Findings suggest that the best suited model for the task is the NN model with an accuracy of 87% reached, an improvement to the accuracy reached in other papers.

**Keywords:** music**,** audio, features, automation, cross-validation, models, accuracy

Music Genre Classification

MSBA 315: MAchine Learning

Karim Hazimeh, Elsy Hobeika, Amal Nahle, Anthony-Joe Zougheib

1

**Introduction**

**Hello, our project is about music genre classification**

Today, the world is seeing an explosive growth of songs that are released every single day. Thousands and thousands of songs are becoming available due to the easy access to music platforms by artists and singers, even amateur ones. So how will music streaming and purchasing services such as Spotify and Anghami manage to keep up with classifying the genre of every newly released songs? Here comes the role of **automated** music genre classification which will help users identify music they like. These streaming services can significantly benefit from automatic classification of the genre since it will result in decreased labor time and costs, and will enhance the user’s experience by providing a better music recommendation system, which in turn will lead to customer loyalty and higher revenues for the company.

Music genre is the most important component in describing music content (Aucouturier & Pachet, 2003). it contains members sharing common characteristics that can be differentiated from other music genres such as rhythm, melody, harmony, and instrumentation (Tsai *et al,* 2010)*.* Audio signals generally carry information about music genre (Scaringella *et al.,* 2006). Automating music classification will be done through algorithms which try to model these signals through their statistical distribution of the short time features (Panagakis *et al,* 2008*).*

*Our objective is to improve the accuracy…*

The aim of this paper is to develop different machine learning models that can be used for automatic music genre classification. The authors of this paper will try to improve the accuracy reached by other authors on the chosen dataset, detailed in further sections of the report.

The first section of the report will highlight related work conducted and published by other authors. The second part will guide the reader through the methodology followed by the authors. The section that follows will emphasize the results and discuss them, while the last part will include conclusions and recommendations.

**Literature Review**

A music genre classification approach typically includes processing the audio files of a dataset, extracting the descriptive features of each file, and using pattern recognition algorithms to classify the music genre (Panagakis *et al,* 2008)*.* Features that can be used for classification are: audio features (such as rhythmic, pitch-content, and timber texture features), textual features (song lyrics and titles), visual (album covers), and multimodal (combines several of the aforementioned features) (Lansdown, 2019). This paper will focus solely on audio features since available resources (such as audio libraries on Python) and most studies focus on an audio-based approach. As for the most frequently employed classifiers, they include Support Vector Machines (SVM), Nearest Neighbor (NN), Non-negative Matrix factorization (NMF), linear discriminant analysis, Gaussian Mixture Models (Panagakis *et al).*, Neural Network, Random Forest, and Gradient Boosting Machine.

Tzanetakis and Cook (2002) use KNN and Gaussian mixture models to perform music genre classification for audio features on the GTZAN dataset and obtain an overall accuracy of 61%.

In a prominent study, Dong (2018) inspired by the human music genre detection uses Convolutional Neural Network (CNN) to model features extracted from spectrograms on the GTZAN dataset. LibROSA library is used to transform the wavelength in a mel spectrogram. Each audio file consisting of 30 seconds was split into 3-seconds files, and the data was split into training and test sets through cross-validation. Different accuracies were reached across different genres, with the highest accuracy of 70% obtained for classical and blues music genres.

In another study, Choi *et al.* (2017) compare CRNN and CNN with three variations models on the Million-Song dataset. Results showed that CRNN outperforms all of the latter across all music genres with an accuracy of above 80% for all genres.

In a study conducted by Lansdown (2019), the author uses the FMA (Free Music Archive) dataset and creates two smaller versions of the latter by under-sampling and oversampling techniques for genre-balancing to classify music genre. Forty-one features were extracted using LibROSA audio analysis library and 13 using Aubio which first requires conversion from mp3 to WAV format. Prior to the classification phase, two sets of analysis were performed. The first compared the average value of each feature for each genre. The second compared and ranked the features by their F-value to see which would be the most relevant in classification. Four classifiers were employed: SVM (different kernel types were tested), neural network, random forest, and a gradient boosting machine. A five-fold cross validation on the training data was used in all experiments. The under-sampled dataset performed better than the oversampled one. In terms of accuracy on the under-sampled dataset, the ranking of the models from best to worst is: SVM (68%), Gradient Boosting Machine (66%), Random Forest (63%), and Neural Network (60%). Results indicate that the SVM is the a most suited algorithm to the task, with Hip-Hop being the most accurately classified music genre.

This study uses the GTZAN dataset where an overall accuracy of 82.5% was reached by Bergstra *et al* (2006), around 78.5% by Li *et al*. (2003), and 78.2% by Panagakis *et al* (2008). The report focuses on improving the aforementioned accuracy on the GTZAN dataset using different techniques like cross-validation, dimensionality reduction, extraction of audio features, and classifiers like SVM and CNN.

**Methodology**

This section will detail the methodology steps followed which include data selection, data splitting, feature extraction, pre-processing techniques, and machine learning models, while also highlighting the best model which resulted in the highest model accuracy.

**Data Selection**

Choosing a dataset is an important part of all machine learning projects. Often, there are several datasets available that could potentially be suitable for the project, each having its pros and cons. Hence, deciding on a dataset can play a significant role in defining the project course. In this present study, the authors decided on using the GTZAN dataset which is detailed in Tzanetakis (2002). This dataset is a known resource when it comes to the field of Music Information Retrieval (MIR). It is also available to anyone online for free. The data includes 10 files each containing 100 – 30 seconds audio song excerpts, with each file representing a specific music genre (blues, classical, disco, country, jazz, hip-hop, pop, metal, rock, and reggae). Hence, the total number of audio files is 1000 overall. One of the advantages of this data set is that it comes organized in folders, each one containing a specific genre, making it easy to navigate. Moreover, although the dataset can be considered relatively small, it has a balanced nature making it appealing for any machine learning project.

**Feature Description**

For music classification purposes, audio features can be categorized into three essential types of audio features: rhythmic features, timbral texture features, and pitch content features (Panagakis *et al,* 2008*)*. This section will outline the features that will be extracted for this present study and that will potentially be selected for use in machine learning models. These include the following:

* MFCCs
* Zero-Crossing Rate
* Spectral Spread
* Chroma Features
* Average Tempo
* Mel Spectrogram

MFCCs

MFCCs are timbral (“color” or “quality” of the sound) texture features considered the most commonly used in audio classification, including ambient noise and speech recognition. Ordinarily, 13 – 30 MFCCs can be extracted, several coefficients which can be decided by the researcher. In this present study, 30 MFCCs are first extracted, and the optimal number which gives the best accuracy is later selected. (Lansdown, 2019).

This study uses for each audio file the mean value of MFCCs (i.e. there is only one MFCC value for MFCC6, which is the average value of all MFCC6 over all the windows of the signal). This means for each audio file, there are 30 values each representing 1 MFCC from MFCC1 till MFCC30.

Zero Crossing Rate

Zero crossing rate is also a timbral texture feature commonly used in music genre classification. It is defined as the rate at which the signal changes from negative to positive and vice-versa. This refers to the amount of time the signal’s amplitude passes through zero in a specific time interval. This feature can be used to measure the “smoothness” of the audio signal or the existence of a percussive sound. This factor is relevant for music genre classification since some genres use percussive sounds more than others (Lansdown, 2019).

Spectral Spread

Spectral Spread, or spectral centroid, is another timbral texture feature used for audio files. It represents the center of gravity of the signal’s spectrum. Perceptually, it refers to the brightness of the signal’s sound. It also describes the variance of the rate map around the centroid. Noisy sounds have a higher spectral spread, while spectrums that are tightly concentrated around the centroid typically have lower spectral spreads. Excerpts from electronic music, for example, are more widely spread around the centroid than Jazz excerpts (Lansdown, 2019).

Chroma Features

Chroma-based features are pitch content features that are associated with notes played throughout a song. There exist numerous techniques to detect musical pitch, with newer ones that focus on pitch detection of the polyphonic sequences. Audio analysis packages can estimate the intensity at which the notes are present within the audio file. It can also estimate how they change over time. This is an important factor in music classification since different music genres lean towards also different key signatures. Usually, the note that is most frequently played in a song is its root note (Lansdown, 2019).

Tempo

Musical tempo is a rhythmic feature used in music classification. Even though a song might have several changes in tempo through its duration, most music has a fixed number of beats/minutes. Since there is a rhythmic emphasis on each beat of a certain bar, estimating the number of beats/minute is typically straightforward as long as these emphases are detected. Naturally, different genres are played at a different speed, making the tempo feature suitable for genre classification. Speed metal, for example, is played at higher tempos compared to doom metal.

Mel Spectrogram

The mel spectrogram can be considered a visual representation of the audio signal. It refers to how the frequencies spectrum change over time. Here, the Fourier Transform is performed on overlapping windowed sections of the signal to obtain the spectrogram or the visual representation of the spectrum frequencies over time. The frequencies are mapped to the mel scale (which is a pitch measure) since humans are not able to perceive frequencies on a linear scale, making equal distances in the pitch sound distant equally for the human ear (Roberts, 2020).

**Data Splitting**

As previously mentioned, the original data set contains 1000 - 30 seconds audio files. However, in order to see whether increasing the data would improve the developed models, the data was divided into 3 seconds audio files, increasing the overall amount of data. The following procedure in this section and in all the upcoming sections of the report was conducted for both the 30 seconds dataset and the 3 seconds dataset individually.

The data was split into train and tests set (80-20%). Then, the training dataset was split into train and validation sets through cross-validation (10- folds). This decision of using cross-validation was based on the Literature Review and was due to the many advantages of K-fold cross-validation such as obtaining a better estimate of the out-of-sample accuracy, and a more efficient usage of data (since each observation is used in both validation and training). The features and the label (the music genre) were also specified (X and Y respectively).

**Feature Extraction**

Feature Extraction as Numerical Values

Tracks need to be represented numerically so that machine learning models can be later applied. From each file, audio features were extracted and represented by their means, which is the average over the entire audio file/ across all the windows, along with their variance, totaling an overall of 83 features. The features were all extracted using Python’s LibROSA library for audio analysis. The extracted features include: timbral texture, pitch content, and rhythmic features. The selection of the features extracted was based on the conducted Literature Review where all the extracted features were shown to be very important for music genre classification since they embody sonic information that are relevant to the classification. The rhythmic features included “Tempo”, while the pitch content features included “chroma\_stft”, “chroma\_cqt”, and “chroma\_cens”. The other features, such as 30 MFCCs part of the timbral texture audio features. The numerical features were extracted and exported into Excel files. Feature extraction as numerical values was followed on both datasets: the one containing the 30-seconds audio files, and that containing the 3-seconds audio files.

Feature Extraction as Images

Mel spectrograms were extracted as images and used in the models separately from the 83 other features (meaning they were included alone in the model). When extracting spectrograms as images, things like framing of the picture and x and y-labels and axis were removed since they are considered noise. the spectrograms were extracted as *.png* files.

Feature extraction as images was only done on the data set containing the 30 seconds audio files, due to computational time-limitations.

**Pre-processing**

Before data extraction, one audio file was disregarded because the file could not be read, which means that the final data was composed of 999 audio files instead of 1000. Pre-processing steps included checking for duplicates in the extracted features. There turned out to be no duplicates out of all the data. Finally, a Min-Max scaler was applied on the features.

**Feature Selection**

This section concerns numerical features which were extracted and imported to Excel and not the extracted images.

In an attempt to further increase accuracy and achieve dimensionality reduction, backward selection was conducted to check whether dropping a certain feature will negatively impact accuracy. Whenever the accuracy dropped when dropping a feature, the feature was kept; otherwise, it was dropped. While conducting the backward selection, every feature seemed to be important, meaning all features which were extracted were kept. Moreover, since as mentioned earlier in the *Feature Description* section, 13 – 30 MFCCs can ordinarily be used with the number of coefficients decided by the researcher. The authors tried experimenting with lowering the number of MFCCs used from 30 to 25, 20, and 15. However, the accuracy dropped at each trial, meaning all 30 MFCCs which were extracted showed to be useful. The authors also experimented with dropping all variances which were extracted, only keeping the average value of each feature. Nevertheless, this also led to a decrease in accuracy. Hence, all features which were initially extracted were selected to be used in the models.

**Machine Learning Algorithms Using Numerical Features**

Six models were developed using the numerical features including: Gaussian Naïve Bayes, Stochastic Gradient Descent, KNN, SVM, Logistic Regression, and Random Forest. These models were chosen based on the conducted Literature Review.

The *RandomizedSearchCv* function in Python was used to develop all mentioned models, with a 10-fold cross-validation (for training and validation) and 5 to 30 iterations for each model. Different parameters were used for each model, with the range for the parameters and choice of parameters based on common practices for each model. Hence, **optimization** of the models was done through the RandomizedSearchCv function which helps obtain the best model parameters that results in the best model accuracy after going through the many specified iterations.

The following table shows the best parameters obtained for each model, along with the model accuracy (on validation) on the **30-seconds** dataset.

Table 1- Models Comparison on 30 seconds data

|  |  |  |
| --- | --- | --- |
| Model | Best parameters | Accuracy |
| Gaussian Naïve Bayes | - | 57% |
| SVM | C=10,  decision\_function\_shape='ovo'  degree=1 | 75% |
| Logistic Regression | Solver= ‘saga’  Penalty= ‘l2’  C= 100 | 72% |
| Random Forest | Max\_depth= 70  N\_estimators= 200 | 70% |
| KNN | Leaf\_size= 29  P=1 | 67% |

The following table shows the best parameters obtained for each model, along with the model accuracy on the **3-seconds** dataset.

Table 2- Models comparison on 3 seconds data

|  |  |  |
| --- | --- | --- |
| Model | Best parameters | Accuracy |
| Gaussian Naïve Bayes | - | 52% |
| Stochastic Gradient Descent | Loss =’log’  Max\_iter = 3000  Penalty= ‘l1’ | 69% |
| SVM | C=100  decision\_function\_shape='ovo'  degree=0 | 87% |
| Logistic Regression | Solver = ‘liblinear’ | 56% |
| Random Forest | Max\_depth = 20  N\_estimator = 2000 | 81% |
| KNN | Leaf\_size = 7  N\_neighbors = 3  P = 1 | 89% |

Comparing models accuracies in both datasets, the model with the highest accuracy on validation is the **KNN** model using the 3-seconds dataset with an accuracy of **89%.**

A pipeline was created so that the steps including pre-processing, feature engineering, and the best model (KNN) would be applied easier to the test set. An accuracy of **88%** was reached when the pipeline was applied on the test set.

**Deep Learning**

Convolutional Neural Networks (CNN)

Before talking about the methodologies used in implementing CNN, a new architecture has been introduced during Thirty-sixth International Conference on Machine Learning 2019, *“EfficientNet: Rethinking Model Scaling for Convolutional Neural Network*” which is a novel model scaling method that uses a simple yet highly effective compound coefficient to scale up CNNs in a more structured manner. Unlike conventional approaches that arbitrarily scale network dimensions, such as width, depth, and resolution, this method uniformly scales each dimension with a fixed set of scaling coefficients. Powered by this novel scaling method, a new family of models was developed, called Efficient Nets, which super pass the state-of-the-art accuracy with up to 10x better efficiency (smaller and faster).

For this part, mel spectrograms which were extracted as images from the 30-seconds dataset (an image for each audio file) were used.

The following callbacks were built for the models:

* Early Stopping: whenever the monitored metric stops improving, it stops training.
* ReduceOnPlateau: whenever the monitored metric stops improving, it reduces the learning rate.
* ModelCheckpoint: using model.fit(), it is utilized in combination with training in order to save a model in a certain checkpoint file at an interval. Hence, the model could be later loaded to go forward with training from the saved state.

The first model consisted of 3 convolutional layers, then a dense layer of 128 units with RELU activation. After compiling the model and fitting it with 30 iterations, the model was overfit with 99% train accuracy compared to 56% for validation as shown in Figure 1.

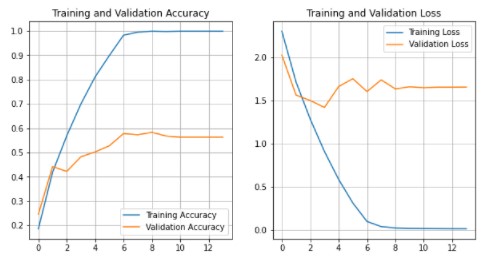
****

Figure 1- Train and validation accuracy and loss

For the second model, an EfficientNetB0 (where a target size/resolution for the images was set to 224) was built with 10 output classes, initialized from scratch and with a dense layer of softmax. However, the problem of overfitting persisted as the results show in Figure 2.

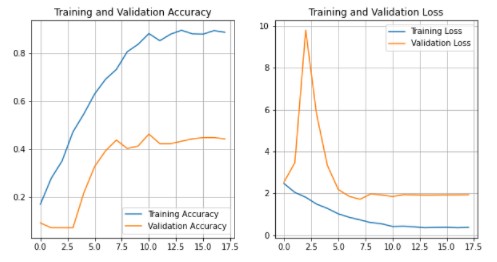


Figure 2- Train and validation accuracy and loss

A common way to reduce overfitting is through using a dropout rate. This was initialized in the third model with pre-trained ImageNet weights, and was fine-tuned in the dataset. The first step to transfer learning is to freeze all layers and train only the top layers.  Although the training accuracy dropped to 66%, the problem of overfitting problem was resolved where the accuracy on the validation set was close to 60%. To further improve the accuracy, the whole layers were unfreezed while the BatchNorm layers were left frozen. Then, we fit the model iterate over 30 epochs. both train and validation accuracies saw an improvement in this model with 71% accuracy for validation and 85% for training, as shown in Figure 3 below.



Figure 3- Train and validation accuracy and loss

Neural Networks (NN)

For this part, only numerical features which were extracted from the 3-seconds dataset were used. This was due to the fact that the 3-seconds dataset resulted in higher accuracies in the machine learning models highlighted in previous sections.

After loading the data, the labels were first updated to indices, hence each label is referred to a unique number ('blues': 0, 'classical': 1, 'country': 2, 'disco': 3 etc.). Then, the data was shuffled using panda’s sample function and resetting the index. The data was divided into target and features variables where target is the label already defined. The data is split into 70% train (6992 records), 20% validation (1978 records), and 10% test (1019 record).

The 3 data sets were scaled using *Standard Scaler* while taking into account data leakage. Three functions were defined prior to model evaluation. The first function is *myCallback* which works similarly to kera’s callback *Early Stopping*, however, the accuracy threshold was defined as 0.94 as a maximum accuracy reach. The second function is *trainModel* which takes model epochs and the optimizer used and fits the model on the train set. What was used were metrics accuracy and loss argument “sparse categorical cross entropy”, which produces a category index of the most likely matching category. The third function will allow the plotting of the results.

Four models were built:

* **Model 1**

The first sequential network constitutes of a dense layer of 256 neurons, the second layer of 128 neurons, the third of 64 neurons, and the fourth dense layer is of 10 neurons. The three layers are activated with RELU where everything less than 0 turns out as 0, while everything higher than 0 turns out as the value itself. The last dense layer is activated with softmax, the latter takes the 10 outputs and normalizes them so that they can add up to 1, ending up as being probabilities. Now considering the highest scoring or probability out of the 10 as the prediction, this will directly correspond to the highest number position.

The model was trained with 70 epochs and the optimizer used is *Adam*. The following figure shows the results of Model 1 with 100% accuracy on training and 87% on validation.

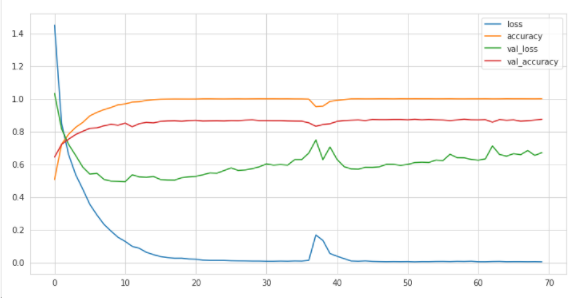


Figure 4- Model 1 results

* **Model 2**

To reduce overfitting, the dropout rate is used in this model, and the dense layers are increased to 5 (the first layer is of 512 neurons while the other layers are similar to Model 1), keeping the last one as softmax activation. The model was trained with 20 epochs and the optimizer used is also *Adam*. The following figure shows the results of Model 2 with 95.5% accuracy on training and 87.8% on validation.

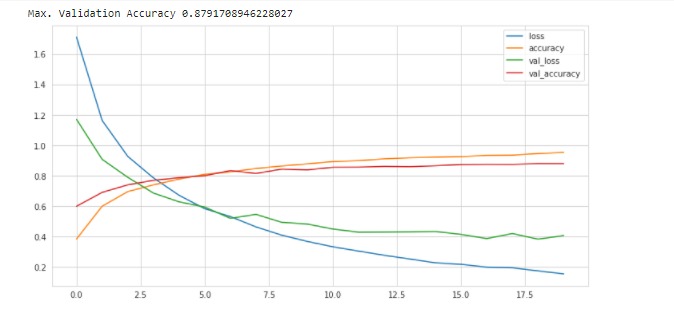


Figure 5- Model 2 results

* **Model 3**

Model 3 is similar to model 2 in terms of dense layers and dropout rate, but here the model is trained with more epochs (150) and optimization was done with *Stochastic Gradient Descent*. The following figure shows the result of Model 3 with 88.8% accuracy on training and 86.3% on validation.

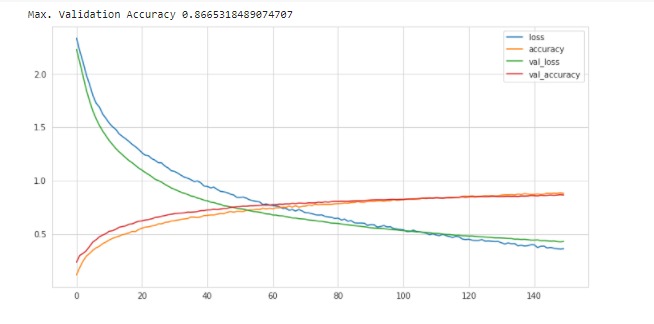


Figure 6- Model 3 results

The following table displays the results of the neural networks along with their architecture.

Table 3- Results comparison

|  |  |  |
| --- | --- | --- |
| Model | Architecture | Validation score |
| CNN  (30s) | 3 Convolutional layers  RELU activation  128 units; 30 iterations | 56% |
| NN  (3s) | 3 dense: 256,128 ,64  Softmax; RELU  Dropout; 70 epochs  Adam optimizer | 87% |
| NN  (3s) | 5 dense: 512, 256,128 ,64  Softmax; RELU  Dropout; 20 epochs  Adam optimizer | 87.8% |
| NN  (3s) | 5 dense: 512, 256,128 ,64  Softmax; RELU  Dropout; 150 epochs  Stochastic Gradient Descent optimizer | 86.3% |

**Results and Discussions**

The following table compares the best models obtained throughout the study.

Table 4- Models comparison

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Architecture | Train Accuracy | Test Accuracy |
| Logistic Regression 30 seconds | Solver= ‘saga’; Penalty= ‘l2’  C= 100 | 88% | 73% |
| KNN  On 3s | Leaf\_size = 7; N\_neighbors = 3  P = 1 | 99% | 88% |
| CNN  On 30s mel-spectrogram | 3 Convolutional layers,  dense layer of 128 RELU activation.  30 iterations, | 85% | 71% |
| NN | 5 dense : 512, 256,128 ,64  Softmax  RELU  dropout  150 epochs  Stochastic Gradient Descent optimizer | 96% | 87% |

After testing the models above using different methods of feature extraction, our findings conclude that the **NN** with the previously specified architecture is the best model for music genre classification with an accuracy of **87%** on the test set. This accuracy is an improvement compared to Dong’s (2018) work which resulted in a 70% accuracy using CNN on the same dataset. From there can be concluded the fact that extracting music features (MFCC, Zero Crossing Rate, Spectral Spread, Chroma Features, and Average Tempo) numerically and using NN is a better strategy for music genre classification.

**Conclusions & Recommendation**

This study compared different machine learning models to see which is the best suited to the task of classifying music genre. The model that showed the most promising results is the NN model with an accuracy of 87% reached on the 3-second data set compared to an overall accuracy of 82.5% reached by Bergstra *et al* (2006), around 78.5% by Li *et al*. (2003), and 78.2% by Panagakis *et al* (2008). The classification was based on the extraction and selection of 83 features from each audio file as numerical values, and a mel spectrogram as an image from each. The models were applied on both 30-seconds and 3-seconds audio datasets.

Concerning limitations, the authors mainly faced time limitations: computational time limitations such as not being able to do a lot of iterations while using *RandomizedSearchCv*; not having enough time to try out libraries such as Aubio to conduct feature extraction; not extracting additional features.

Since only one rhythmic feature (“tempo”) was extracted during the study, future work can emphasize on similar features to see whether the classification accuracy can be further improved. Future work could also include using, other than audio features, textual features which include song lyrics and titles, visual ones which include album covers, and even multimodal which combine many of the aforementioned features. Also, working on a larger dataset or even combining the used data set could be an option. Generally, music genre classification is still a great yet interesting challenge for both a business and an academic institution, and there is much room for extra study and analysis.

The authors of this paper recommend companies to start investing in machine learning to automate music classification for all new releases. This will help in providing a better service for customers especially the niche customers who struggle to find their taste since all the focus is on the main genres. A better service means retaining customers, increasing customer loyalty, and targeting more segments. This will result in more revenues for the company, along with a decrease in labor cost for labelling and human errors. Machine learning is a science that is now contributing in many industries and is still growing where there is room for better model performance.

**References**

Choi, K., Fazekas, G., Sandler, M., and Cho, K. Convolutional recurrent neural networks for music classification. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2392–2396, March 2017. doi: 10.1109/ICASSP.2017.7952585.

G. Tzanetakis and P. Cook, “Musical genre classification of audio signals,” IEEE Transactions on Speech and Audio Processing, Vol. 10, 2002, pp. 293-302.

J. J. Aucouturier and F. Pachet, “Representing musical genre: A state of the art,” Journal of New Music Research, pp. 83–93, 2003.

Lansdown, Bryn. Machine Learning for Music Genre Classification. 2019

Mingwen Dong. Convolutional neural network achieves human-level accuracy in music genre classification. CoRR, abs/1802.09697, 2018

N. Scaringella, G. Zoia, and D. Mlynek, “Automatic genre classification of music content: A survey,” IEEE Signal Processing Magazine, vol. 23, no.2, pp. 133–141, Mar. 2006.

Tsai, W.H., Bao, D.F. Clustering music recordings based on genres. In Information Science and Applications (ICISA), 2010 International Conference on (pp. 1-5).IEEE., 2010.

Y. Panagakis, C. Kotropoulos, and G. R. Arce. Music genre classification via sparse representations of auditory