UNIVERSITY OF LONDON

BSc Computer Science (Machine Learning and Artificial Intelligence).

CM3070 PROJECT
PRELIMINARY PROJECT REPORT

< Music Genre Classification >

Author	:	Evalyn Low Wei Xuan
Student Number	:	190428734
Date of Submission	:	5 Sept 2022
Supervisor	:	Dr. Tarapong Sreenuch

Contents

CHAPTER 1: INTRODUCTION

Motivation

This project focuses on the classification of music genres, which can assist in recommending songs on streaming services such as Spotify, Youtube, etc., as the song recommendation could be inaccurate at times. When recommending music, metadata such as music genres are taken into account. Hence, music genres are one of the key factors that help these music streaming services understand your taste. As a result, a better model could be developed that automates the classification of music genres by creating a more effective automated process of classification, which will then improve music information retrieval systems (Vekriya, Vyas and Dedlhiya, 2020).

Template

The template chosen for this final year project is Machine Learning Template 1: Deep Learning on a Public Dataset. The dataset used for this project will be the GTzan dataset. It consists of 10 genres with 100 tracks in each, for a total of 1000 audio tracks, each with a 30-second duration. It contains the spectrograms of the audio and two CSV files containing the features of the audio tracks. This dataset is placed on Kaggle and the information is available at this link: GTZAN Dataset - Music Genre Classification | Kaggle

Aims

The primary aim of this project is to create a classification model using machine learning techniques. The libraries that will be used for deep learning are TensorFlow, Keras, Sklearn, and various Python libraries, as these are commonly used in the industry. The language used is Python. The coding environment will take place in a Jupyter notebook.

The secondary aim of the project is to see if it is possible to do a soft classification of music genres, as we know that some music has genres that sit on the boundaries between different genres. For example, Ed Sheeran's "South at the Border" has both hip hop and pop genres. Hence, soft classification is important in music genre classification because it allows streaming services' recommendation systems to be more accurate, as well as because users sometimes prefer music from different genres mixed together.

Deliverables

This proposal will include a review of the literature on various cutting-edge machine learning classification methods of music genres, as well as the overall project design, model implementation, and model evaluation. It will also include a comparison of all of the models that were created during the course of the project. This will determine whether the model developed is effective and accurate enough to be deployed as an API for an application or system.

CHAPTER 2: LITERATURE REVIEW

Literature Review 1: Content-Based Classification

The source is reliable and it can be trusted as the paper is published on https://www.researchgate.net/. Researchers and scientists publish their work on this website. Based on this fact, the source of this paper is reliable (ResearchGate | Find and share research, 2022).

A literature review is necessary because the objective of this paper—the automation of music genre classification—perfectly complements the objective of my project.

The paper proposed using Daubechies Wavelet Coefficient Histograms (DWCH) as a new feature to classify music genres. This research paper uses various machine learning techniques such as Support Vector Machines and Linear Discriminant Analysis. The paper was split into two different sections: feature extraction and multi-class classification, which is the section on deep learning (Li, Ogihara and Li, 2003).

The feature they wanted to focus on was the content-based acoustic features, which are classified into timbral texture features, rhythmic content features, and pitch content features. These features include the STFT, Spectral Centroid, Spectral Rolloff, Mel-FrquencyCoefficient, etc. As they have researched in the past, the highest accuracy for these features is 61%. (Tzanetakis and Cook, 2002). With this accuracy as a baseline, they wanted to use the feature based on the wavelet coefficient histograms, which is the DWCH, along with some machine learning techniques, and as a result, they managed to raise the accuracy to 80 % (Li, Ogihara and Li, 2003).

In addition, they also researched several machine learning techniques, such as support vector machines, Gaussian mixture models, and k nearest neighbour. They experimented by using a single feature, then with a mixture of features, and finally with the DWCHs. They concluded that SVMs are the best classifiers for content-based classification. Through the experiments, they also concluded that using the DWCHs feature further improves the accuracy of all the methods that they used. Taken from the paper, these are the results [Appendix Figure 1].

Reading the paper motivates me to extract the DWCHs feature from my dataset and observe if the accuracy of my model increases. As seen from the result, they can hit an average of 70% in the classification of music genres (Li, Ogihara and Li, 2003).

Literature Review 2: Finding the perfect network for music genre classification

The source for this paper is the International Conference of Computational Science. Here, researchers from the fields of mathematics, computer science, and various application areas discuss problems and solutions in their respective fields to shape the future direction of research. Hence, this paper is reliable (International Conference on Computational Science: ICCS 2022).

This research is all about using neural networks and deep learning models to classify music genres.A literature review is necessary because the objective of this paper—the automation of music genre classification—perfectly complements the objective of my project. This paper states that it experiments with the usage of various neural networks, such as convolution neural networks, convolutional recurrent neural networks with 1D and 2D convolutions, and recurrent neural networks with Long Short Term Memory Cells (Kostrzewa, Kaminski and Poland, 2021).

The experiment is split into 2 parts, where the first part is that they observe the results by classifying the genres using several deep learning models before comparing the results. The second part is where they would combine the deep learning models, mix the models, and make an ensemble either by using a fully connected layer or by voting to classify the genres (Kostrzewa, Kaminski, and Poland ,2021).

The evaluation metrics for this project are accuracy, precision, f1, and recall. With the evaluation metric settled, they went on and experimented with different models. The result that came out was surprising and interesting as the LSTM model was one of the worst-performing models. This is interesting as the LSTM cells should consider the sequence of the signal and use it to their advantage, and therefore should not be the worst performing model. After testing each of the models, they combine different network architects into different ensembles and test the classification (Kostrzewa, Kaminski, and Poland ,2021).

After the experiment, it was concluded that the accuracy of the ensemble using the voting technique was higher than the ensemble using the technique of fully connected layers. By using the confusion matrix to visualize the results from the best model, which is the voting ensemble, we can see that an almost 80% accuracy rate for the genre hip hop was achieved and that the accuracy for the next four genres is between 60% and 70%, producing a satisfactory result[Appendix figure 2]. In conclusion, the researchers think that this experiment must be expanded to conduct further experiments using deep network architectures.

After reading this research paper, using deep network architectures could be a possible way to help me increase the classification accuracy. Thus, the goal of using a neural network to improve classification accuracy. (Kostrzewa, Kaminski, and Poland, 2021).

Literature Review 3: Deep learning-based music genre classification using spectrogram

This next literature review is adapted from the Social Science Research Network. This network is devoted to the worldwide distribution of research papers. It also comprises several specialized research networks. This makes the research paper a reliable and trustable source(SSRN, 2022).

A literature review is necessary because the objective of this paper—the automation of music genre classification—perfectly complements the objective of my project. This project uses the audio spectrogram generated from the audio dataset to classify them using CNN.

Feature extraction and model building are the two main parts of the project. The first part is feature extraction. The extracted features are Zero Crossing Rate, Spectral Centroid, Spectral Roll-off, Spectral Bandwidth, Spectral Contrast, and MFCCs. These features will be passed through the Librosa Python Library to perform the feature extraction using different techniques. The researchers will then use a function called mel_spectogram, which will take in the spectrogram features. The researcher then stored these in a Numpy array consisting of the spectrogram. After the extraction of the features, data augmentation is performed. This is to increase the amount of training data (M.K and S, 2021).

Moving on to the second part, we will now discuss model building. The Keras library was used to build a 2D CNN, which will be demonstrated in the implementation chapter. This model takes a binary version of the spectrogram as the input. Once the model has been built with the necessary optimizer and loss function, they will then train the model for 200 epochs using TensorFlow. After gathering the results, they achieved a 94% accuracy rate for the model. They compared their results with other methods done by other researchers, as you can see in the table [Appendix Figure 3]. This allowed them to achieve the highest accuracy (M.K and S, 2021).

After reading the paper, the motivation for using CNN as the model and spectrograms as training input became clear. In the paper, they also included the development of a recommendation system. From this, the possible idea of using RNN as a model to improve the accuracy instead of CNN is raised. (M.K and S, 2021).

Literature Review 4: Music Genre Classification using GMM

The last research paper is adapted from the International Research Journal of Engineering and Technology, making the paper very reliable as the source where researchers in Engineering and Technology go to publish their discussions and findings (International Research Journal of Engineering and Technology 2022).

This paper is similar to the project that I will be working on. It focuses on using the Tempogram as one of the features extracted and uses the Gaussian mixture model to classify the music genres. A literature review is necessary because the objective of this paper—the automation of music genre classification—perfectly complements the objective of my project. (Thiruvengatanadhan, 2018).

Like the aforementioned paper, this project is split into 2 parts, feature extraction and model building. The features that they are going to extract are temporal features. It captures the local tempo and the beat characteristic of the music signal. They have extracted and windowed short segmented frames in the preprocessing stage of the project. They then computed the novelty curve, as shown below [Appendix Figure 4]. This curve will then indicate which peaks represent the note onset values. The Fourier tempogram is then calculated, and finally, the histogram is computed for each frame, resulting in 12-dimensional feature vectors.

Once the features have been extracted, it is time to build the model. The model that they have chosen is the GMM (Gaussian Mixture Model). GMM is used in this project to classify different audio classes, and the classifier itself is a parametric classifier. When the model is trained, it can be used to predict which class it belongs to. The definition of the model is that it is a probabilistic model where it assumes all the data points are generated from a finite number of Gaussian distributions. The input for the model would be the feature vector. The researcher has chosen a mixture of 2,5,10 mixture models where it uses the Expectation maximization (EM) to fit all the parameters into the model. It proved to be very effective as the accuracy has hit up to 94.9 %. It was also concluded that the best-performing Gaussian model was achieved with 10 Gaussian mixtures (Thiruvengatanadhan, 2018).

A temporal feature is a way to characterize the audio content. Together with the tempogram feature, they used the GMM learning algorithm to classify the genre by learning from the training data. It then uses the proposed method where they use the EM approach to fit the parameters, attaining an accuracy of 94%. After reading the research paper, it appears that the GMM can be one of the possible ways to classify the music genre. However, there are a lot of parameters to fit into the model for it to achieve such a high level of accuracy (Thiruvengatanadhan, 2018).

CHAPTER 3: PROJECT DESIGN

Project design is where the domain and users will be defined. Additionally, it will outline the general project structure, the model's design rationale, the technologies employed, and how the model will be tested and assessed.The gannt chart for the project will also be covered in this chapter.

Domain

The domain of the project belongs to music classification. The automation of classification is a fundamental component of Music Information Retrieval (MIR). In the past, the genre of the audio was normally embedded in the metadata of the audio file and was manually typed into the metadata. Hence, with the automated classification of genres, it can reduce human errors, making it a valuable asset to the MIR system (Automated Music Genre Detection and Classification using Machine Learning, 2020).

Moreover, the classification of genres provides a basis for creating and assessing features for any form of content-based analysis of the signal in the music audio (An intelligent music genre analysis using feature extraction and classification using deep learning techniques, 2022)

User

The target consumers for this project are streaming services and also users of those streaming services. The automation of genre classification is a fundamental component of the MIR system, which includes a recommendation system. With an improved recommendation system, the streaming service could increase its loyal customer base.

In addition to that, having a better recommendation system also gives the company the ability to interact with its customers (Underwood, 2019). For example, the streaming service could survey to see what music genres are most popular, so they could bring in more artists of the same genre.

Justification on design

The reason for designing an automated music genre classifier is that it can provide an improvement to the recommendation system in streaming services. With an improvement to the recommendation system, it will help the streaming services maintain a loyal customer base where there is more user feedback and find out which type of music genre attracts them most, so that the service could bring in more artists of similar genres.

Additionally, the advantage of using machine learning techniques to automate the process is that we can reduce human error as the machine learning algorithm will learn the characteristics of the categories and separate data accurately. With machine learning introduced, there are a lot of applicational uses where the MIR can be improved. It can separate the audio by pattern recognition in music sources and recognize the instrument used in the audio. With this, it can be further expanded to the field of speech recognition, where it can recognize speech in a noisy environment.

The overall structure of the project

The overall project structure would refer to the methodology that will take place during this project. At this stage, the methodology for building my model will be introduced. The following is the methodology of my project (AltexSoft, 2018).

The steps taken to create my project would be as follows:

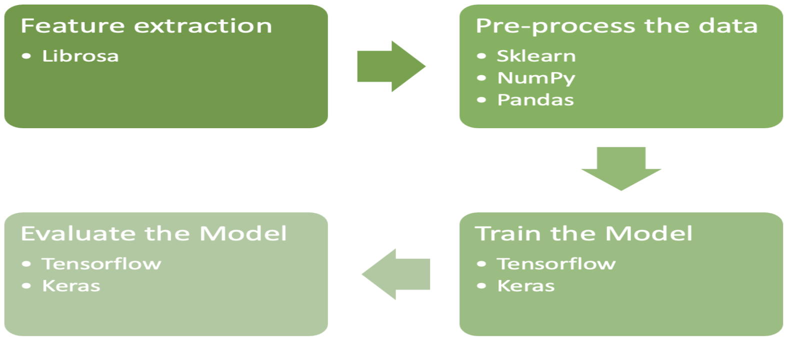
1) Identify the project's purpose
   With this project, we aim to create a machine learning-based classifier to automate music genre classification (AltexSoft, 2018).
2) Dataset preparation and preprocessing
   I found the dataset that I wanted to analyse and work on, which is the GTzan dataset. Looking through the dataset, I can see that there are 3 types of data that we can work on: audio files, spectrograms, and 2 CSV files of the features that are already extracted from the audio (AltexSoft, 2018).
   As for the preprocessing step, I would need to clean and format the data. An example of cleaning the data is to see if there are any discrepancies in the data, like null values or anomalies. The purpose of this stage is that we can convert the data into features so that it can fit into the later stage, which is the modelling stage (AltexSoft, 2018).
3) Dataset Splitting
   Once the data has been cleaned and formatted in a way that fits the model, we would need to split the data into train and test sets. The training set is the set that is going to train the model, whereas the test set is to test the model and see its accuracy. Within the train set, we will split it even further as a validation set. The validation set is when training, we can tweak the model hyperparameter to achieve higher accuracy (AltexSoft, 2018).
4) Modelling
   This stage is where I will need to train numerous models to find the best model. This is where we will feed the model the training set and let the model learn from it. Once the model is trained, it will then be validated by the validation set to see if the accuracy has hit the target. If the model does not hit the target, I will need to tune the hyperparameter to make it pass the accuracy that I have in place. Once the model has been tested and validated, it will then be tested again with the test set to gain the final accuracy. This will result in the final model (AltexSoft, 2018).
5) Model Deployment
   Once the model has been tested, it is now ready to be deployed. This final model will then be translated from a high-level language, which is Python in this case, to a low-level language such as Java and C/C++. After translating, its performance will need to be measured by A/B testing. Another method of deployment would be the state of the art in machine learning as a service or AI as a service. The deployment of these would be automated, and the results could be delivered through REST APIs. (AltexSoft, 2018).

Identification of technologies used

The technology project that will be a part of this project will build a machine learning-based classifier that will automate the classification of music genres used:

1) Programming language: Python 3.9
2) Environment: Jupyter Notebook
3) Libraries Used: TensorFlow, Keras, Pandas, Numpy, Sklearn, Librosa

I chose these technologies as they are dependable and help achieve the functions of my algorithm. The reason the Jupyter notebook was chosen as my environment is that it can be displayed in a step-by-step manner so that other people can understand the thought process being shown. The libraries that I have chosen are used in various machine learning projects and Python projects. Below is where these libraries would be used.



My Work Plan

Over the course of this semester, I have broken the project into smaller tasks. I have created a Gannt Chart [Appendix Figure 5]. To keep up with the task on hand, I have come up with this chart to keep me in check for the entire project timeline. This will enable me to calculate the amount of time I would need to devote to the component itself.

I have 2 milestones that I need to complete this project. One would be the Preliminary Report on June 27 and the Final Report on September 6.

My project is divided into numerous milestones, as can be seen in [Appendix Figure 5]. With this timetable in hand, I could keep track of my progress. The start date and due date are stated.

Test and Evaluate

In a classification problem, accuracy would be the most important feature. Therefore, by evaluating the accuracy score, we can see if the model is indeed useful and effective when predicting the genre. I would also pay attention to whether the test's accuracy can reach a benchmark of 61%. If the model does not hit it, I would consider it to be performing poorly and falling short of the requirements for a final model. This accuracy is based on Tzanetakis and

Cook's (2002) published research paper (Tzanetakis and Cook, 2002). At the same time, the model must be able to pass the naive model’s accuracy of 10%.

I will be assessing the model against the test set as soon as I am satisfied with the validation accuracy. I would have to rethink the model architecture and start over if the test accuracy did not at least match the first developed model's accuracy.
Since I'm using a machine learning template, the classification report with precision, recall, and f1-score would be printed out. Here is more information on these terms:

1) Precision
   This metric is in charge of determining how frequently the model predicts the right outcome (Gavrilova, 2020).
2) Loss
   This metric is in charge of describing the proportion of inaccurate predictions the model makes (Gavrilova 2020).
3) Recall
   This metric measures how many total relevant results the model correctly categorizes (Gavrilova 2020).
4) F1 Score
   This metric is the weighted average of precision and recall. The best value is 1 and the worst is 0.0 (Mardani, 2020).

Seeing the score for the evaluation metrics, we can see if the model is indeed effective. We can focus primarily on the f-score by looking at these metrics. This can be seen in the Evaluation part of the Report.

CHAPTER 4: IMPLEMENTATION

Description of the solution

In this chapter, I will be introducing my implemented models. In addition to the 2D Convolutional Neural Network (CNN), which is trained on the MFCCs features, I also included K nearest neighbour (KNN), Support Vector Machine (SVM), and Sequential Model, which are trained on the structured data that was supplied. These models are trained based on the Gtzan dataset, which I have mentioned in the above chapters. **The project codes are available here.**

Using the CSV of the audio features, I was able to come up with three models: SVM, KNN, and a sequential model, which is a simple network. As mentioned above, the base accuracy is 61% based on research (Tzanetakis and Cook, 2002). In my prototype, I used the CSV that consists of the features of each 30-second audio and resulted in a validation accuracy of 73.7% for the SVM, 68.125% for KNN, and 63.1% for the Sequential Model **[Appendix Table 1].** Seeing that the data is small, I decided to use another CSV file that comprises all the audio features for 3 seconds. Essentially, a 30-second audio file is split into 3-second audio snippets, giving the three models 10,000 lines of data to train with. However, one of the audio files was corrupted, resulting in 9990 lines of data. I was able to explore the various features and do the preprocessing using the CSV data.

Firstly, using the sklearn library, I managed to preprocess the data by scaling the values using the module'sklearn.preprocessing' with MinMax Scaler. The three models I used are SVM, KNN and a sequential model. After scaling the data, it is time to prepare the data. This is where we make sure the data is suitable to be fed into the model. The features are the X data, while the y data would be the label (genre) of the music.

Secondly, I split my data into an 80% training set and a 20% test set. I split it further into a validation set (20%) within the training set. Before training my model on the whole dataset, I will use the validation set to validate my model.

Finally, modelling is performed using the three models SVM, KNN, and a Sequential Model. The first model, SVM, is a supervised learning algorithm that could help me classify the structured data in the CSV. The second model, KNN, is a pattern recognition algorithm that is used in many music genre classification problems as they look for a similar pattern before assuming that they are from the same class. The last model is a sequential model, a simple neural network consisting of a fully connected dense layer. The following is a description of how I trained the models and my thought process while implementing the solution.

After training it countless times with the validation set, I came to the conclusion that the model was overfitting and the results were erratic. Therefore, I decided to use a technique called GridSearch CV, which will be explained in the next part of the report, to find the perfect hyperparameter for both KNN and SVM. After being content with the results of the validation set, the entire training set was trained using the same methodology, and the results were then compared to the validation set. This produced test accuracy values for the KNN model and the SVM model of 91.0% and 92.8%, respectively, with the best hyperparameters [Appendix Figure 6].

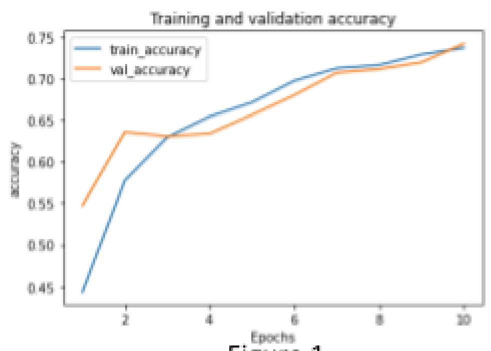


Figure 1

Using the same preprocessed CSV data, I was also able to create a neural network. As a baseline, I used the generated graph. [Figure 1], and the validation accuracy (74.1%) of the first model. The graph demonstrates that the results produced fluctuate while the validation accuracy continues to rise. As a result, I'd have to keep adjusting hyperparameters like epoch count, layer count, and neuron count in each layer. After adjusting the hyperparameters manually, I decided to automate the process with the Keras tuner library. This will be explained in more detail in the report's later section.

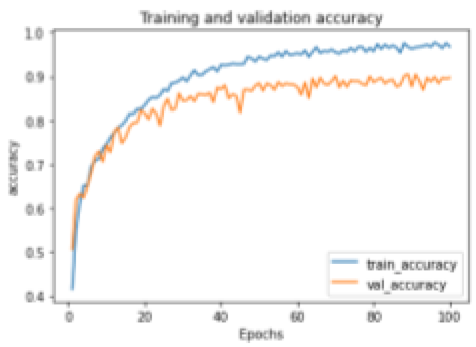


Figure 2

Using this method, I obtained a validation accuracy of 89.7% and the generated graph is stable [Figure 2]. As a result, using the adam optimizer and sparse_categorial_crossentrophy and a callback to prevent overfitting, I was able to train the entire training set with the best hyperparameters **[Appendix Figure 7]**. With the final model yielding a 91.1% test accuracy and a 97% training accuracy. I generated the genres' classification report. **[Appendix Figure 8].** Despite the positive results of these models, I'd like to use the audio wav files provided in the dataset to see if they can be classified. After extensive research, I discovered that one of the key characteristics in determining the genre of music is the MFCCs of the audio. Using this feature, I decided to use the 2D CNN model as it can learn the patterns that are translation invariant and spatial hierarchies (Chollet, 2018).

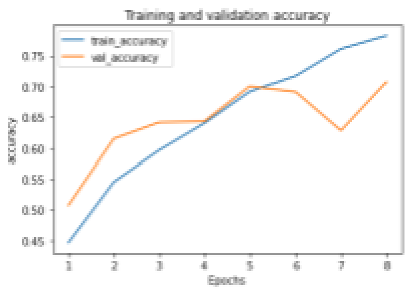


Figure 3

I first extracted the MFCCs out of the audio and then stored them in a JSON file so that they could be passed as input into the neural network. I decided to split the audio into 13 segments because my dataset is too small and each segment has 13 MFCC coefficients. This is done to increase the dataset size in order to achieve greater accuracy. Therefore, we ended up with 12985 rows of data to train on. Using the same methodology as the previous models, I first loaded the data into the JSON file and then store it in an array. The data will be divided in the same way as it was for the models aforementioned above.

Similarly to the neural network, I used the first model's accuracy (79.74%) and produced a graph [Figure 3] as my baseline. The first model is produced using an Adam optimizer, a categorial_crossentrophy loss function, 30 iterations, and a callback to avoid overfitting. In **[Appendix Figure 9]**, it came to an end at epoch 8 in order to stop overfitting. Hence, I'll tune my model using the Keras Tuner moving forward. The best model architect,**[Appendix Figure 10]** had a validation accuracy of 76.66% after I looked into a number of architects to be inserted into the Keras tuner. From the graph generated [Figure 4], we can see that at epoch 5 of this model, the model is starting to stabilise. Therefore, I would train this model on the entire training set using 5 epochs, Adam optimiser, a categorical crossentrophy loss function, and the best model architect**[Appendix Figure 10]**, and then compare its performance to the test set. As a result, the training accuracy was 96.7% and the test accuracy was 81.6% **[Appendix Figure 11]**, yielding the highest accuracy. This model, in my opinion, is the best and should be regarded as a final model when compared to the first model I made using this input data.

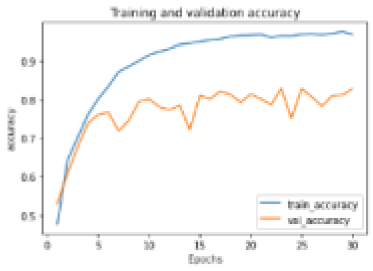


Figure 4

Explanation of Techniques used in the Solution

Grid Search CV, the Keras tuner, and callbacks. These methods are employed in machine learning to avoid overfitting. MFCCs are a crucial feature to be extracted whenever we are working on the audio signal. It is saved as a JSON file to prevent any data loss (Sanket Doshi, 2018).

1. Grid Search Cross Validation
   Grid search is a method that calculates the best results by combining all of the hyperparameters I've specified. As you can see, I'll be using this technique on both the KNN and SVM models in this case. This method incorporates cross-validation and grid search (Shah, 2021). This allowed the model to produce accurate predictions.
2. Keras Tuner
   Its function is to automate the process of hyperparameter tuning of the hyper model. Its function is to find the ideal combination of hyperparameters to achieve the best validation accuracy. In our situation, I wanted to maximize the validation accuracy, so the tuner found the best hypermodel to achieve this.
   A hyperband is an optimized version of a random search. It begins by running the configuration for two epochs, then chooses the best configuration and continues to fine-tune it. I'm initializing the tuner with a hyperband, which is an optimization of the random search algorithm (Conol, 2020).
3. Callback
   It is to allow model training to end once model performance on the validation set stops advancing. I would like the model to stop training if the validation loss performance keeps dropping after 3 epochs, which could indicate overfitting. This method is demonstrated when I am training the 2D CNN model (Vijay, 2019) and the sequential model. It also allows me to see if the validation loss has improved since the previous epochs. If the model improves, it will be saved as a hd5 file. It'll then load the model and train it on the entire training set. This is to ensure that only the best model is evaluated.

CHAPTER 5: EVALUATION OF THE SOLUTION

In this chapter, I am going to evaluate my solutions. In machine learning, we assess the model by assessing the testing set, which is a dataset that has not been seen during the model training phase. Additionally, we evaluate evaluation metrics like the model's accuracy and the classification report. As a recap of the previous chapter, I used various techniques to prevent overfitting in each of these solutions, including increasing the dataset, using the Keras tuner library to identify the ideal hyperparameter for the model architecture, and adding callbacks that would cause the model to stop being trained if the validation loss had decreased, and employing Grid Search Cross-Validation. Furthermore, by dividing the training set into validation and training sets, I was able to detect overfitting early on and refine the model before determining the test set results [Table 1].

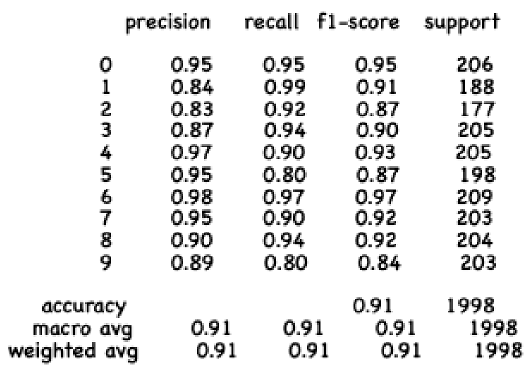
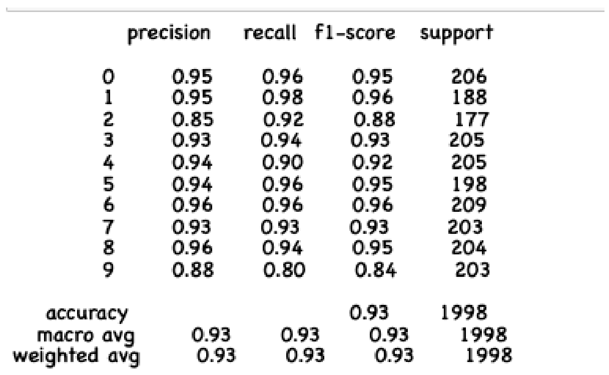
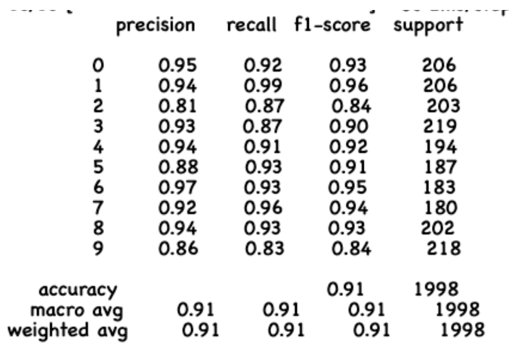
Model	Training Accuracy	Test Accuracy
Naive Model (Probability of blind guessing)	10%	
SVM	90.1%	92.8%
KNN	89.8%	91.0%
Sequential Model	97.6%	91.1%
2D CNN trained with MFCCs	96.7%	81.6%

Table 1: Results of Models

As seen from the results, every model has provided a satisfactory result, which exceeded the benchmark set (61%). Additionally, we can see that accuracy for all models has significantly increased when compared to the 10% naive model. We can see that classical machine learning techniques such as KNN and SVM could hit a test accuracy of at least 90%. These models' accuracy has undoubtedly increased from the prototype's average of 70% [Appendix Table 1]. Additionally, the sequential model’s accuracy has been improved significantly from the base model of 70%.

As you can see from the table above, the 2D CNN that was trained with the MFCCs feature has also achieved a test accuracy of 81.6%. This is a significant improvement from the base model that I had implemented, whereas the accuracy could only hit a validation accuracy of 70% [Appendix Table 1]. We can see that this model has a lower test accuracy than the rest of the models. This, I think, is because I'm only honing in on one audio feature, as opposed to the CSV data, which contains a variety of audio features that could be used to precisely identify the genre. In addition to that, I feel that even though I have expanded my dataset, it is still small in size.

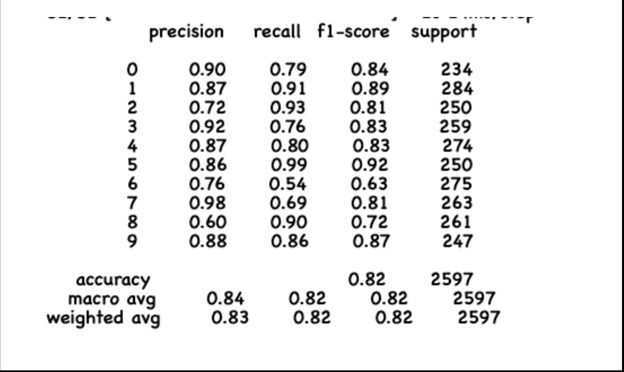
In addition to noting the test accuracy, I have printed out the classification report for each model. [Table 2]The models are able to classify the genres quite accurately and precisely when evaluating the test set.

Model	Classification Report
KNN	
SVM	
Sequential	

```
              precision    recall  f1-score   support

           0       0.95      0.95      0.95       206
           1       0.84      0.99      0.91       188
           2       0.83      0.92      0.87       177
           3       0.87      0.94      0.90       205
           4       0.97      0.90      0.93       205
           5       0.95      0.80      0.87       198
           6       0.98      0.97      0.97       209
           7       0.95      0.90      0.92       203
           8       0.90      0.94      0.92       204
           9       0.89      0.80      0.84       203

    accuracy                           0.91      1998
   macro avg       0.91      0.91      0.91      1998
weighted avg       0.91      0.91      0.91      1998
```

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95       206
           1       0.95      0.98      0.96       188
           2       0.85      0.92      0.88       177
           3       0.93      0.94      0.93       205
           4       0.94      0.90      0.92       205
           5       0.94      0.96      0.95       198
           6       0.96      0.96      0.96       209
           7       0.93      0.93      0.93       203
           8       0.96      0.94      0.95       204
           9       0.88      0.80      0.84       203

    accuracy                           0.93      1998
   macro avg       0.93      0.93      0.93      1998
weighted avg       0.93      0.93      0.93      1998
```

```
              precision    recall  f1-score   support

           0       0.95      0.92      0.93       206
           1       0.94      0.99      0.96       206
           2       0.81      0.87      0.84       203
           3       0.93      0.87      0.90       219
           4       0.94      0.91      0.92       194
           5       0.88      0.93      0.91       187
           6       0.97      0.93      0.95       183
           7       0.92      0.96      0.94       180
           8       0.94      0.93      0.93       202
           9       0.86      0.83      0.84       218

    accuracy                           0.91      1998
   macro avg       0.91      0.91      0.91      1998
weighted avg       0.91      0.91      0.91      1998
```

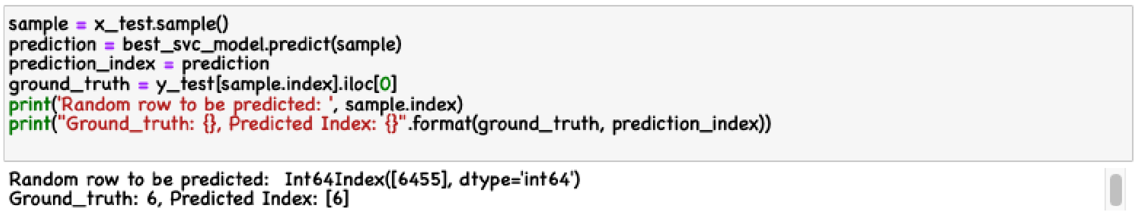
2D CNN	

```
              precision    recall  f1-score   support

           0       0.90      0.79      0.84       234
           1       0.87      0.91      0.89       284
           2       0.72      0.93      0.81       250
           3       0.92      0.76      0.83       259
           4       0.87      0.80      0.83       274
           5       0.86      0.99      0.92       250
           6       0.76      0.54      0.63       275
           7       0.98      0.69      0.81       263
           8       0.60      0.90      0.72       261
           9       0.88      0.86      0.87       247

    accuracy                           0.82      2597
   macro avg       0.84      0.82      0.82      2597
weighted avg       0.83      0.82      0.82      2597
```

Table 2 : Classification Report

As you can see from the above reports, I notice that KNN has the highest accuracy and f1 score when classifying the genres. When compared to the other models, the scores for the sequential model and the SVM model are quite comparable, but the scores for the 2D CNN model were marginally lower than the others. Because of this, my chosen model and solution will be KNN.

Since I'll be using SVM as my final model, I chose a random sample from the test set and used it to make predictions to show the model's accuracy [Figure 4]. The predicted index is what my SVM model predicted as the result, whereas the ground truth represents the genre index as it is actually supposed to be.



```
sample = x_test.sample()
prediction = best_svc_model.predict(sample)
prediction_index = prediction
ground_truth = y_test[sample.index].iloc[0]
print('Random row to be predicted: ', sample.index)
print("Ground_truth: {}, Predicted Index: {}".format(ground_truth, prediction_index))
```

```
Random row to be predicted:  Int64Index([6455], dtype='int64')
Ground_truth: 6, Predicted Index: [6]
```

Figure 4: Prediction

Given that the 2D CNN model achieves lower accuracy than the other models, there are some potential extensions to this project. Instead of saving the numbers into the JSON file, melspectograms of the audio will be generated and trained on it. Since CNN is a well-known neural network for image training, it could achieve accuracy comparable to other models. Hence, this could be a further improvement to my project.

CHAPTER 6: CONCLUSION

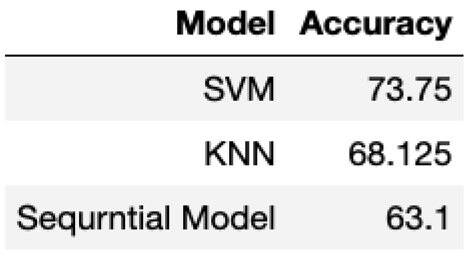
By building a variety of classifiers, we can see that the project's goals have been demonstrated to be attainable and that the outcomes have been satisfactory despite potential dataset limitations of a file being corrupted during download and the dataset being too small to be trained on. However, looking at the results of the classifiers, it can be concluded that the project of classifying the musical genre using machine learning and deep learning techniques has been proven to be quite successful. The codes can be seen on my GitHub repository.

By evaluating the models using the test set, we can see that the models are achieving adequate results. Looking at the results, the sequential model, KNN, and SVM models achieved a test accuracy of 91.1%, while the 2D CNN model averaged 81.6%. [Table 2]. With KNN being the most accurate and effective model, it was chosen as the final model solution. The model was then used to make predictions, which demonstrated its effectiveness and accuracy [Figure 4].

While these classifiers have produced good results, there is always room for improvement. In this study, the only input training data used were the given CSV features and the mfccs saved in a JSON file. Therefore, to improve the classifier's overall quality, future research might focus on other input data, which includes utilizing other audio features such as mel spectrograms. Moreover, the model's deployment into an API so that it can be integrated with web- or mobile-based applications.

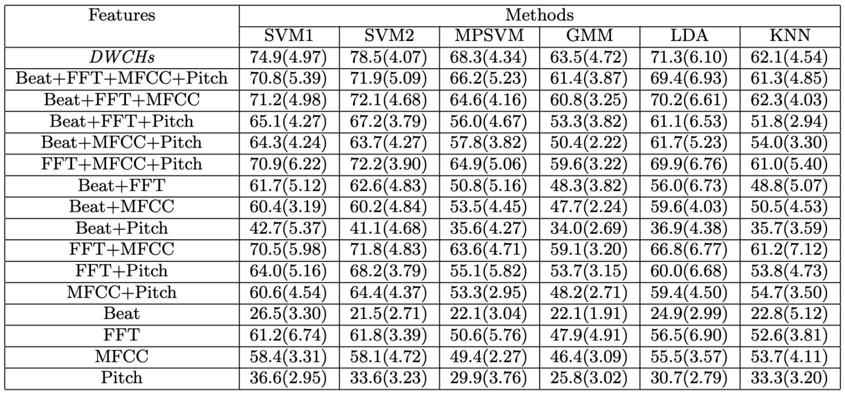
The investigation of machine learning and deep learning methods may lead to the application of these methods in other fields, such as playlist creation and music production. It can also be used in different fields, like speech recognition.

APPENDICES



Model	Accuracy
SVM	73.75
KNN	68.125
Sequrntial Model	63.1

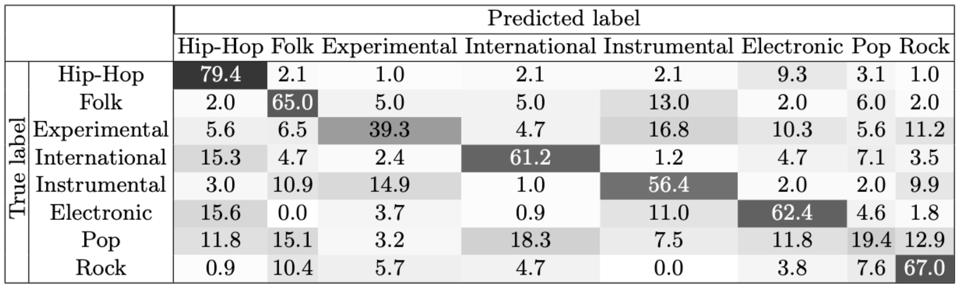
Table 1



Features	Methods					
	SVM1	SVM2	MPSVM	GMM	LDA	KNN
DWCHs	74.9(4.97)	78.5(4.07)	68.3(4.34)	63.5(4.72)	71.3(6.10)	62.1(4.54)
Beat+FFT+MFCC+Pitch	70.8(5.39)	71.9(5.09)	66.2(5.23)	61.4(3.87)	69.4(6.93)	61.3(4.85)
Beat+FFT+MFCC	71.2(4.98)	72.1(4.68)	64.6(4.16)	60.8(3.25)	70.2(6.61)	62.3(4.03)
Beat+FFT+Pitch	65.1(4.27)	67.2(3.79)	56.0(4.67)	53.3(3.82)	61.1(6.53)	51.8(2.94)
Beat+MFCC+Pitch	64.3(4.24)	63.7(4.27)	57.8(3.82)	50.4(2.22)	61.7(5.23)	54.0(3.30)
FFT+MFCC+Pitch	70.9(6.22)	72.2(3.90)	64.9(5.06)	59.6(3.22)	69.9(6.76)	61.0(5.40)
Beat+FFT	61.7(5.12)	62.6(4.83)	50.8(5.16)	48.3(3.82)	56.0(6.73)	48.8(5.07)
Beat+MFCC	60.4(3.19)	60.2(4.84)	53.5(4.45)	47.7(2.24)	59.6(4.03)	50.5(4.53)
Beat+Pitch	42.7(5.37)	41.1(4.68)	35.6(4.27)	34.0(2.69)	36.9(4.38)	35.7(3.59)
FFT+MFCC	70.5(5.98)	71.8(4.83)	63.6(4.71)	59.1(3.20)	66.8(6.77)	61.2(7.12)
FFT+Pitch	64.0(5.16)	68.2(3.79)	55.1(5.82)	53.7(3.15)	60.0(6.68)	53.8(4.73)
MFCC+Pitch	60.6(4.54)	64.4(4.37)	53.3(2.95)	48.2(2.71)	59.4(4.50)	54.7(3.50)
Beat	26.5(3.30)	21.5(2.71)	22.1(3.04)	22.1(1.91)	24.9(2.99)	22.8(5.12)
FFT	61.2(6.74)	61.8(3.39)	50.6(5.76)	47.9(4.91)	56.5(6.90)	52.6(3.81)
MFCC	58.4(3.31)	58.1(4.72)	49.4(2.27)	46.4(3.09)	55.5(3.57)	53.7(4.11)
Pitch	36.6(2.95)	33.6(3.23)	29.9(3.76)	25.8(3.02)	30.7(2.79)	33.3(3.20)

Figure 1

Table 3. Confusion matrix for the best model – Ensemble 1 with Voting [in %].



		Predicted label							
		Hip-Hop	Folk	Experimental	International	Instrumental	Electronic	Pop	Rock
True label	Hip-Hop	79.4	2.1	1.0	2.1	2.1	9.3	3.1	1.0
	Folk	2.0	65.0	5.0	5.0	13.0	2.0	6.0	2.0
	Experimental	5.6	6.5	39.3	4.7	16.8	10.3	5.6	11.2
	International	15.3	4.7	2.4	61.2	1.2	4.7	7.1	3.5
	Instrumental	3.0	10.9	14.9	1.0	56.4	2.0	2.0	9.9
	Electronic	15.6	0.0	3.7	0.9	11.0	62.4	4.6	1.8
	Pop	11.8	15.1	3.2	18.3	7.5	11.8	19.4	12.9
	Rock	0.9	10.4	5.7	4.7	0.0	3.8	7.6	67.0

Figure 2

Table 1
Performance results on GTZAN dataset

Methods	Features	Accuracy
Proposed system	STFT	94%
Music feature maps with CNN[8]	STFT	91%
Deep attention based model[10]	STFT	90%
Parallel recurrent convolutional neural[11]	STFT	91%
Texture selection technique[13]	STFT	88.8%
Bottom-up broadcast neural network[14]	STFT	93.9%

Figure 3

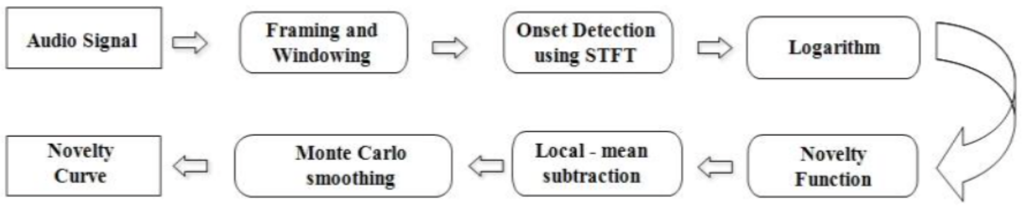
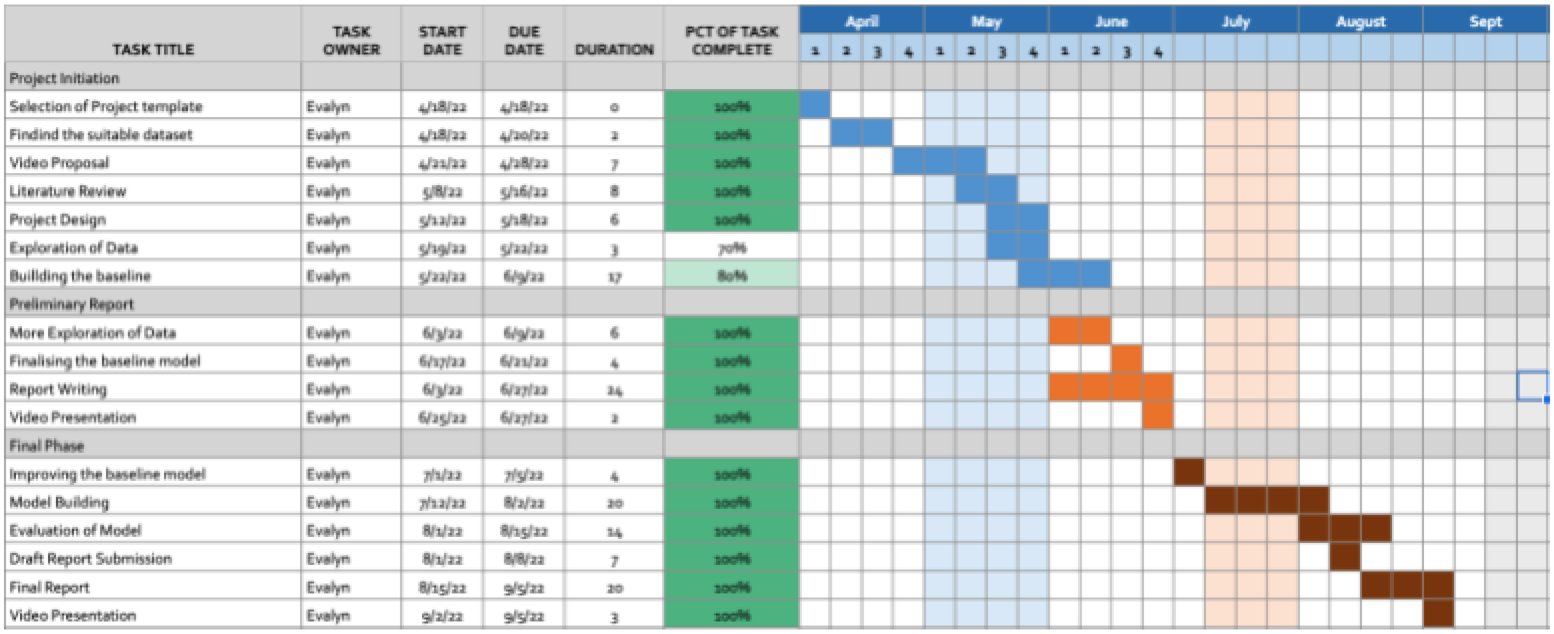


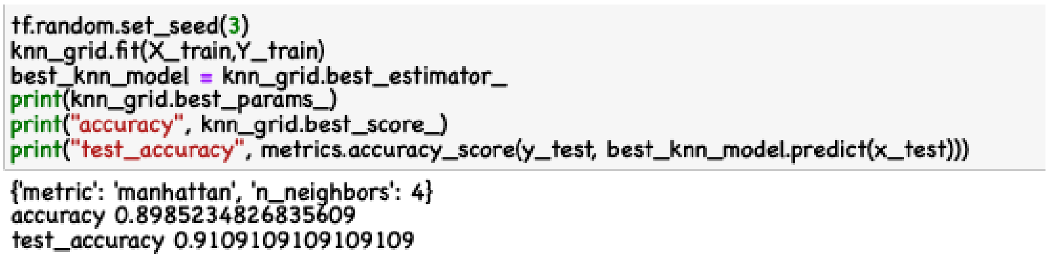
Figure 4

Figure 5 - Gannt Chart



TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE	April				May				June				July				August				Sept			
						1	2	3	4	1	2	3	4	1	2	3	4												
Project Initiation																													
Selection of Project template	Evalyn	4/18/22	4/18/22	0	100%	■																							
Findind the suitable dataset	Evalyn	4/18/22	4/20/22	2	100%		■	■																					
Video Proposal	Evalyn	4/21/22	4/28/22	7	100%				■	■	■																		
Literature Review	Evalyn	5/8/22	5/16/22	8	100%						■	■																	
Project Design	Evalyn	5/12/22	5/18/22	6	100%							■	■																
Exploration of Data	Evalyn	5/19/22	5/22/22	3	70%							■	■																
Buillding the baseline	Evalyn	5/22/22	6/9/22	17	80%								■	■	■														
Preliminary Report																													
More Exploration of Data	Evalyn	6/3/22	6/9/22	6	100%									■	■														
Finalising the baseline model	Evalyn	6/17/22	6/21/22	4	100%											■													
Report Writing	Evalyn	6/3/22	6/27/22	24	100%									■	■	■	■												
Video Presentation	Evalyn	6/25/22	6/27/22	2	100%												■												
Final Phase																													
Improving the baseline model	Evalyn	7/1/22	7/5/22	4	100%													■											
Model Building	Evalyn	7/12/22	8/2/22	20	100%														■	■	■	■							
Evaluation of Model	Evalyn	8/1/22	8/15/22	14	100%																	■	■	■					
Draft Report Submission	Evalyn	8/1/22	8/8/22	7	100%																		■						
Final Report	Evalyn	8/15/22	9/5/22	20	100%																			■	■	■			
Video Presentation	Evalyn	9/2/22	9/5/22	3	100%																					■			

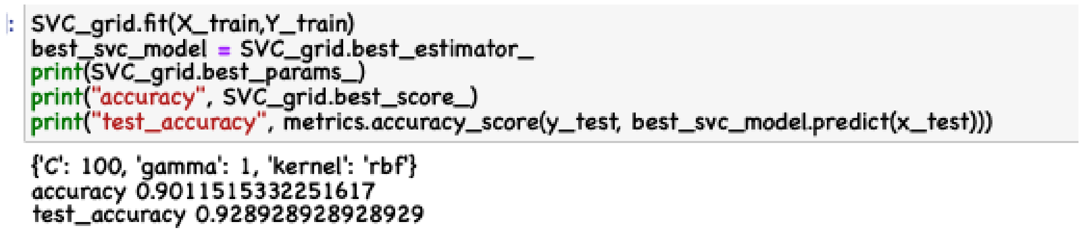
KNN



```
tf.random.set_seed(3)
knn_grid.fit(X_train,Y_train)
best_knn_model = knn_grid.best_estimator_
print(knn_grid.best_params_)
print("accuracy", knn_grid.best_score_)
print("test_accuracy", metrics.accuracy_score(y_test, best_knn_model.predict(x_test)))
```

```
{'metric': 'manhattan', 'n_neighbors': 4}
accuracy 0.8985234826835609
test_accuracy 0.9109109109109109
```

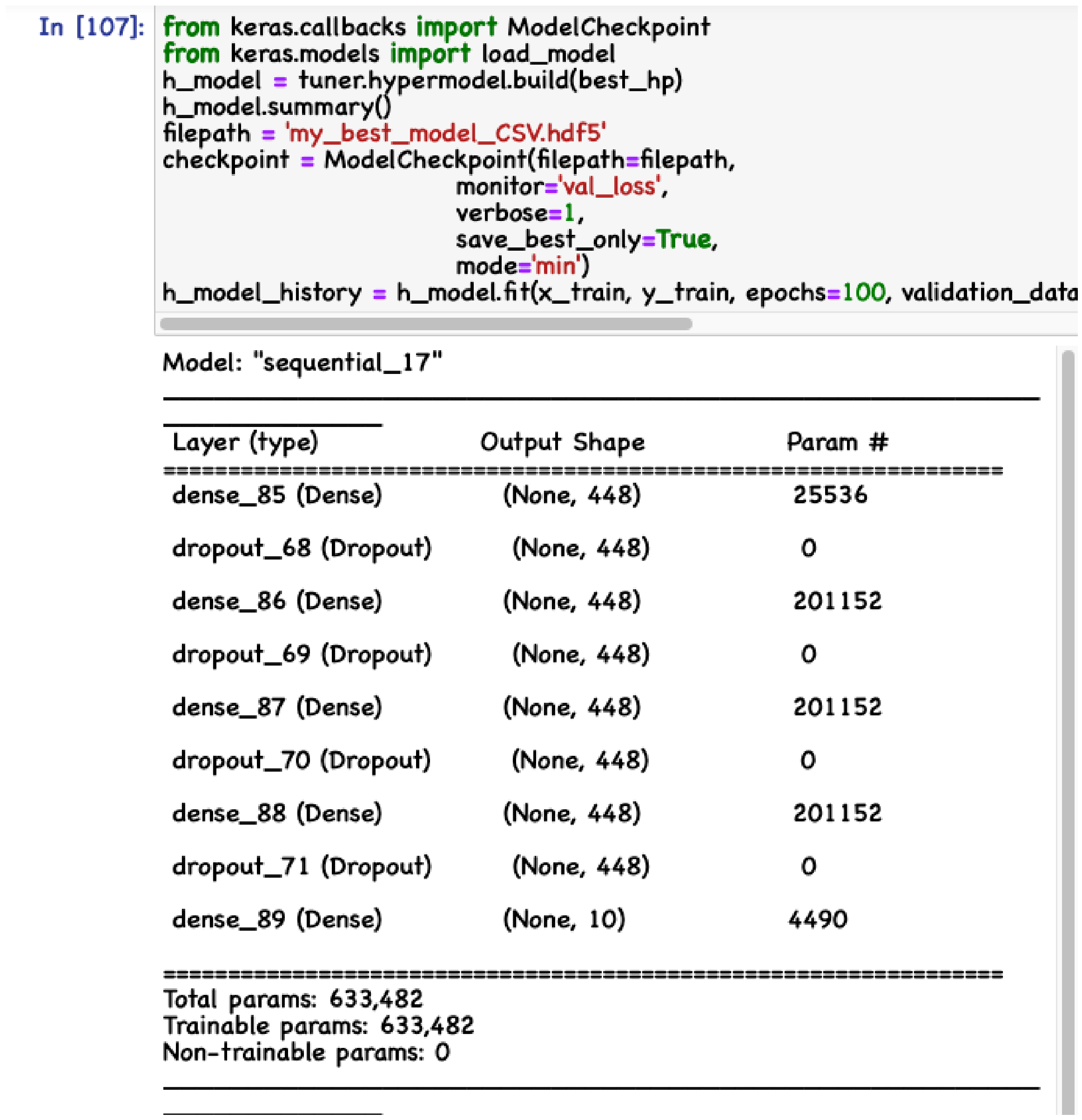
SVM



```
SVC_grid.fit(X_train,Y_train)
best_svc_model = SVC_grid.best_estimator_
print(SVC_grid.best_params_)
print("accuracy", SVC_grid.best_score_)
print("test_accuracy", metrics.accuracy_score(y_test, best_svc_model.predict(x_test)))
```

```
{'C': 100, 'gamma': 1, 'kernel': 'rbf'}
accuracy 0.9011515332251617
test_accuracy 0.928928928928929
```

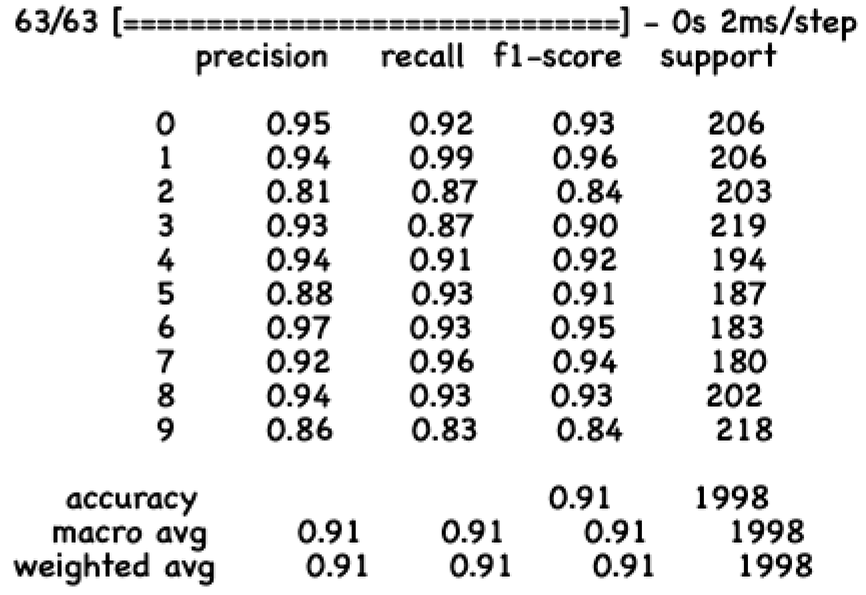
Figure 6 - Best HyperParameter for KNN and SVM , test accuracy



```
In [107]: from keras.callbacks import ModelCheckpoint
          from keras.models import load_model
          h_model = tuner.hypermodel.build(best_hp)
          h_model.summary()
          filepath = 'my_best_model_CSV.hdf5'
          checkpoint = ModelCheckpoint(filepath=filepath,
                                       monitor='val_loss',
                                       verbose=1,
                                       save_best_only=True,
                                       mode='min')
          h_model_history = h_model.fit(x_train, y_train, epochs=100, validation_data
```

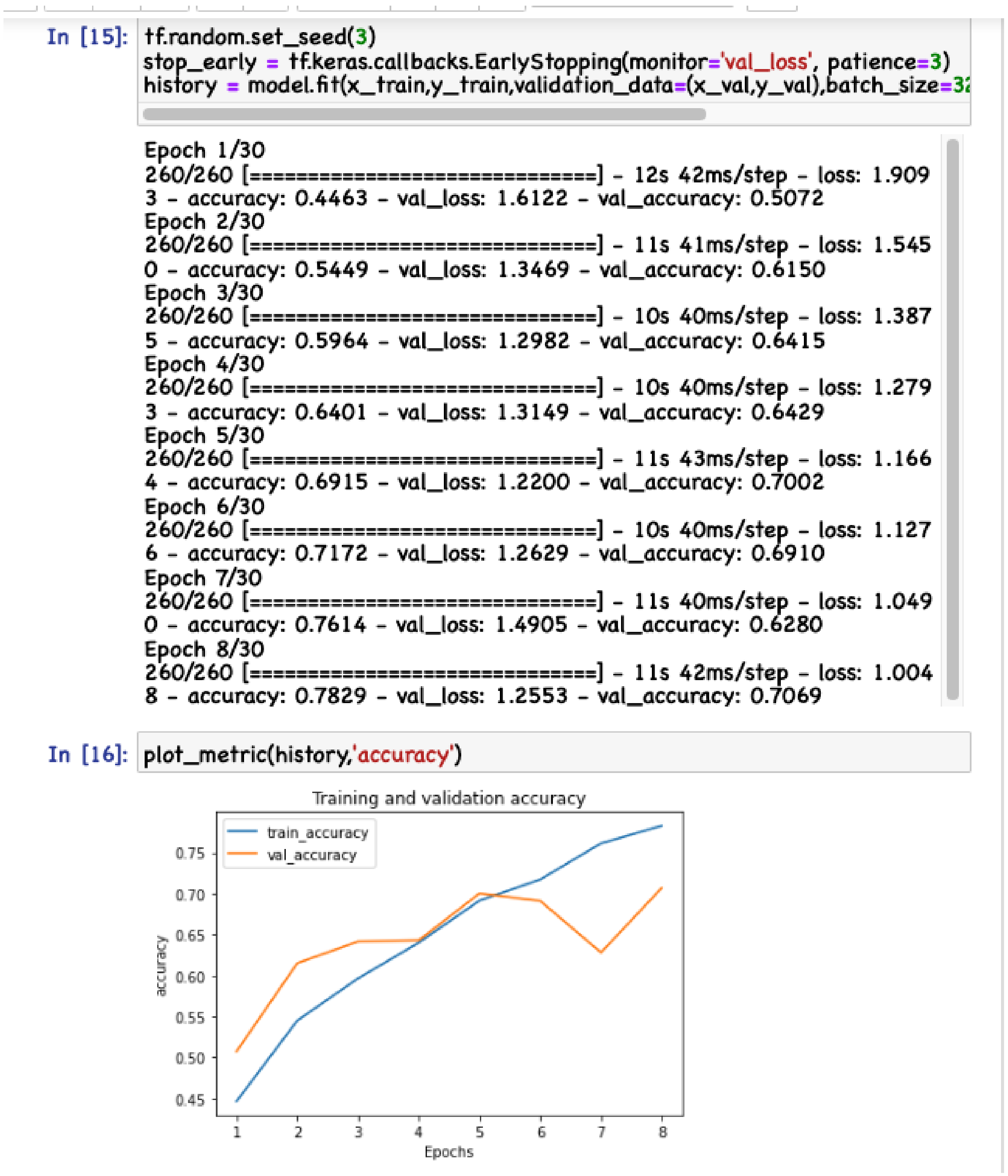
```
Model: "sequential_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_85 (Dense)            (None, 448)               25536

 dropout_68 (Dropout)        (None, 448)               0

 dense_86 (Dense)            (None, 448)               201152

 dropout_69 (Dropout)        (None, 448)               0

 dense_87 (Dense)            (None, 448)               201152

 dropout_70 (Dropout)        (None, 448)               0

 dense_88 (Dense)            (None, 448)               201152

 dropout_71 (Dropout)        (None, 448)               0

 dense_89 (Dense)            (None, 10)                4490

=================================================================
Total params: 633,482
Trainable params: 633,482
Non-trainable params: 0
_________________________________________________________________
```

Figure 7 - Model Architect for Sequential Model



```
63/63 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.95      0.92      0.93       206
           1       0.94      0.99      0.96       206
           2       0.81      0.87      0.84       203
           3       0.93      0.87      0.90       219
           4       0.94      0.91      0.92       194
           5       0.88      0.93      0.91       187
           6       0.97      0.93      0.95       183
           7       0.92      0.96      0.94       180
           8       0.94      0.93      0.93       202
           9       0.86      0.83      0.84       218

    accuracy                           0.91      1998
   macro avg       0.91      0.91      0.91      1998
weighted avg       0.91      0.91      0.91      1998
```

Figure 8 - Classification Report for Sequential Model

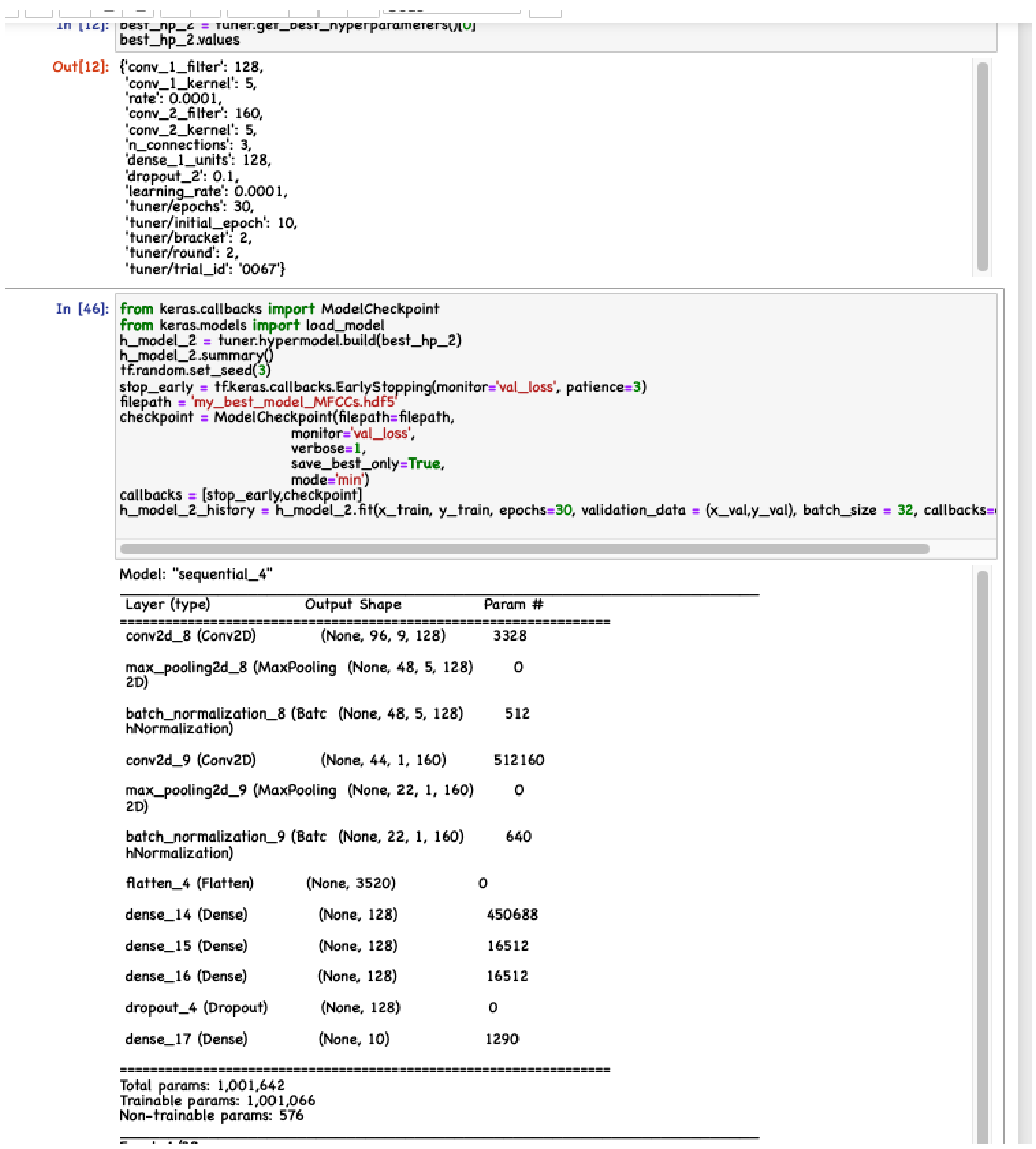


```
In [15]: tf.random.set_seed(3)
         stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)
         history = model.fit(x_train,y_train,validation_data=(x_val,y_val),batch_size=32
```

```
Epoch 1/30
260/260 [==============================] - 12s 42ms/step - loss: 1.909
3 - accuracy: 0.4463 - val_loss: 1.6122 - val_accuracy: 0.5072
Epoch 2/30
260/260 [==============================] - 11s 41ms/step - loss: 1.545
0 - accuracy: 0.5449 - val_loss: 1.3469 - val_accuracy: 0.6150
Epoch 3/30
260/260 [==============================] - 10s 40ms/step - loss: 1.387
5 - accuracy: 0.5964 - val_loss: 1.2982 - val_accuracy: 0.6415
Epoch 4/30
260/260 [==============================] - 10s 40ms/step - loss: 1.279
3 - accuracy: 0.6401 - val_loss: 1.3149 - val_accuracy: 0.6429
Epoch 5/30
260/260 [==============================] - 11s 43ms/step - loss: 1.166
4 - accuracy: 0.6915 - val_loss: 1.2200 - val_accuracy: 0.7002
Epoch 6/30
260/260 [==============================] - 10s 40ms/step - loss: 1.127
6 - accuracy: 0.7172 - val_loss: 1.2629 - val_accuracy: 0.6910
Epoch 7/30
260/260 [==============================] - 11s 40ms/step - loss: 1.049
0 - accuracy: 0.7614 - val_loss: 1.4905 - val_accuracy: 0.6280
Epoch 8/30
260/260 [==============================] - 11s 42ms/step - loss: 1.004
8 - accuracy: 0.7829 - val_loss: 1.2553 - val_accuracy: 0.7069
```

```
In [16]: plot_metric(history,'accuracy')
```

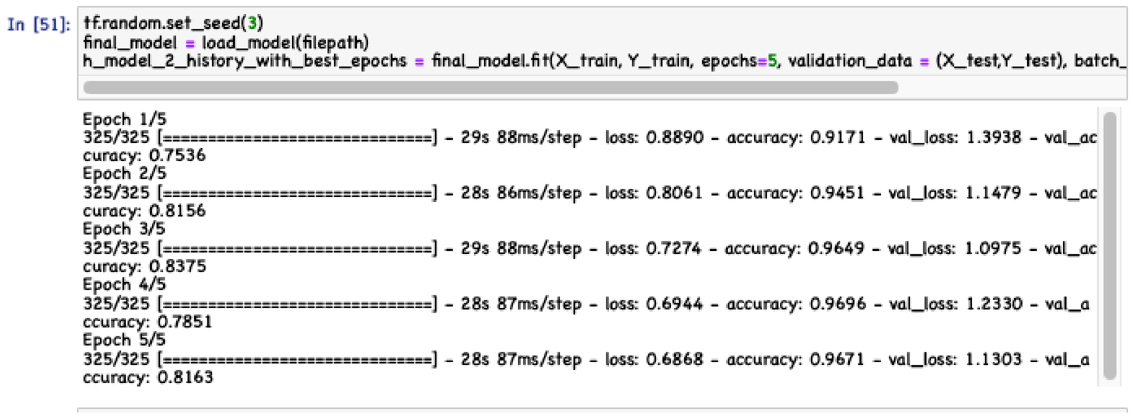
Figure 9- First Model for MFCCs



```
In [12]: best_hp_2 = tuner.get_best_hyperparameters()[0]
         best_hp_2.values

Out[12]: {'conv_1_filter': 128,
          'conv_1_kernel': 5,
          'rate': 0.0001,
          'conv_2_filter': 160,
          'conv_2_kernel': 5,
          'n_connections': 3,
          'dense_1_units': 128,
          'dropout_2': 0.1,
          'learning_rate': 0.0001,
          'tuner/epochs': 30,
          'tuner/initial_epoch': 10,
          'tuner/bracket': 2,
          'tuner/round': 2,
          'tuner/trial_id': '0067'}

In [46]: from keras.callbacks import ModelCheckpoint
         from keras.models import load_model
         h_model_2 = tuner.hypermodel.build(best_hp_2)
         h_model_2.summary()
         tf.random.set_seed(3)
         stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)
         filepath = 'my_best_model_MFCCs.hdf5'
         checkpoint = ModelCheckpoint(filepath=filepath,
                                      monitor='val_loss',
                                      verbose=1,
                                      save_best_only=True,
                                      mode='min')
         callbacks = [stop_early,checkpoint]
         h_model_2_history = h_model_2.fit(x_train, y_train, epochs=30, validation_data = (x_val,y_val), batch_size = 32, callbacks=
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 96, 9, 128)        3328

 max_pooling2d_8 (MaxPooling  (None, 48, 5, 128)       0
 2D)

 batch_normalization_8 (Batc  (None, 48, 5, 128)       512
 hNormalization)

 conv2d_9 (Conv2D)           (None, 44, 1, 160)        512160

 max_pooling2d_9 (MaxPooling  (None, 22, 1, 160)       0
 2D)

 batch_normalization_9 (Batc  (None, 22, 1, 160)       640
 hNormalization)

 flatten_4 (Flatten)         (None, 3520)              0

 dense_14 (Dense)            (None, 128)               450688

 dense_15 (Dense)            (None, 128)               16512

 dense_16 (Dense)            (None, 128)               16512

 dropout_4 (Dropout)         (None, 128)               0

 dense_17 (Dense)            (None, 10)                1290

=================================================================
Total params: 1,001,642
Trainable params: 1,001,066
Non-trainable params: 576
_________________________________________________________________
```

Figure 10 - best CNN model architecture for MFCCs



```
In [51]: tf.random.set_seed(3)
         final_model = load_model(filepath)
         h_model_2_history_with_best_epochs = final_model.fit(X_train, Y_train, epochs=5, validation_data = (X_test,Y_test), batch_
```

```
Epoch 1/5
325/325 [==============================] - 29s 88ms/step - loss: 0.8890 - accuracy: 0.9171 - val_loss: 1.3938 - val_ac
curacy: 0.7536
Epoch 2/5
325/325 [==============================] - 28s 86ms/step - loss: 0.8061 - accuracy: 0.9451 - val_loss: 1.1479 - val_ac
curacy: 0.8156
Epoch 3/5
325/325 [==============================] - 29s 88ms/step - loss: 0.7274 - accuracy: 0.9649 - val_loss: 1.0975 - val_ac
curacy: 0.8375
Epoch 4/5
325/325 [==============================] - 28s 87ms/step - loss: 0.6944 - accuracy: 0.9696 - val_loss: 1.2330 - val_a
ccuracy: 0.7851
Epoch 5/5
325/325 [==============================] - 28s 87ms/step - loss: 0.6868 - accuracy: 0.9671 - val_loss: 1.1303 - val_a
ccuracy: 0.8163
```

Figure 11 - Training accuracy and test accuracy for MFCCs

Presentation slides and Codes link

Presentation slides

https://www.canva.com/design/DAFKyDSrhxk/DxfPflmD2U0kF9jnvgLgZw/view?utm_content=DAFKyDSrhxk&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Codes

The public code repository

https://github.com/evalyn1998/FYP

Dataset used

GTZAN Dataset - Music Genre Classification | Kaggle

References

1. Vekriya, N., Vyas, H. and Dedlhiya, N., 2020. *MUSIC INFORMATION RETRIEVAL AND GENRE CLASSIFICATION USING MACHINE LEARNING TECHNIQUES AND DEEP LEARNING*. [online] Irjet.net. Available at: <https://www.irjet.net/archives/V7/i7/IRJET-V7I7179.pdf>

2. ResearchGate. 2022. *ResearchGate | Find and share research*. [online] Available at: <https://www.researchgate.net/>

3. Li, T., Ogihara, M. and Li, Q., 2003. *A Comparative Study on Content-Based Music Genre Classification*. [online] Research Gate. Available at: <https://www.researchgate.net/publication/221299170_A_Comparative_Study_on_Content-Based_Music_Genre_Classification>.

4. Kostrzewa, D., Kaminski, P. and Poland, R., 2021. *Music genre classification: looking for the Perfect Network⋆*. [online] Iccs-meeting.org. Available at: <https://www.iccs-meeting.org/archive/iccs2021/papers/127420059.pdf>

5. M.K, A. and S, S., 2021. *Deep learning-based music genre classification using spectrogram*. [online] Social Science Research Network. Available at: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3883911#:~:text=Convolutional%20neural%20network%20(CNN)%20is,songs%20into%20their%20music%20genres>.

6. User, S., 2022. *IRJET-International Research Journal of Engineering and Technology*. [online] IRJET-International Research Journal of Engineering and Technology. Available at: <https://www.irjet.net/>.

7. Thiruvengatanadhan, R., 2018. *Music Genre Classification using GMM*. [online] Irjet.net. Available at: <https://www.irjet.net/archives/V5/i10/IRJET-V5I10198.pdf>.

8. Ijesc.org. 2020. *Automated Music Genre Detection and Classification using Machine Learning*. [online] Available at: <https://ijesc.org/upload/863f328a7a3875fecea7de967b0985c9.Automated%20Music%20Genre%20Detection%20and%20Classification%20using%20Machine%20Learning.pdf>.

9. Science Direct. 2022. *An intelligent music genre analysis using feature extraction and classification using deep learning techniques*. [online] Available at: <https://www.sciencedirect.com/science/article/abs/pii/S0045790622002506>.

10. Underwood, C. (2019). *Use Cases of Recommendation Systems in Business – Current Applications and Methods | Emerj*. [online] Emerj. Available at: https://emerj.com/ai-sector-overviews/use-cases-recommendation-systems/

11. AltexSoft (2018). *Machine Learning Project Structure: Stages, Roles, and Tools*. [online] AltexSoft. Available at: https://www.altexsoft.com/blog/datascience/machine-learning-project-structure-stages-roles-and-tools/.

12. Gavrilova, Y. (2020). *Testing Machine Learning Models*. [online] Serokell Software Development Company. Available at: https://serokell.io/blog/machine-learning-testing.

13. Mardani, R.A. (2020). *Practical Machine Learning Tutorial: Part.3 (Model Evaluation-1)*. [online] Medium. Available at: https://towardsdatascience.com/practical-machine-learning-tutorial-part-3-model-evaluation-1-5eefae18ec98.

14. Tzanetakis, G. and Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5), pp.293–302. doi:10.1109/tsa.2002.800560.

15. Chollet, F. (2018). *Deep Learning with Python*. Shelter Island (New York, Estados Unidos): Manning, Cop.

16. Sanket Doshi (2018). *Extract features of Music*. [online] Medium. Available at: https://towardsdatascience.com/extract-features-of-music-75a3f9bc265d.

17. Shah, R. (2021). *GridSearchCV |Tune Hyperparameters with GridSearchCV*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/.
18. Conol, C. (2020). *Hyperparameter Tuning with Keras Tuner*. [online] Medium. Available at: https://towardsdatascience.com/hyperparameter-tuning-with-keras-tuner-283474fbfbe.

19. Vijay, U. (2020). *Early Stopping to avoid overfitting in neural network- Keras*. [online] Medium. Available at: https://medium.com/zero-equals-false/early-stopping-to-avoid-overfitting-in-neural-network-keras-b68c96ed05d9.