

Project Titled

Music Genre Classification using Machine Learning

Submitted

in partial fulfilment of the requirement of
B.Tech(Electronics and Telecommunications)

Report By

Prathmesh Shinde(181090058)

Shubh Ganatra(181090060)

Shubham Agrawal(181090062)

Atharva Wadekar(181090073)



Under the guidance of

DR. D.P. RATHOD

DEPARTMENT OF ELECTRICAL ENGINEERING

VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE

MUMBAI – 400019

(2021-2022)

DECLARATION OF STUDENTS

We declare the work embodied in this project titled “**Music Genre Classification using Machine Learning**” form our contribution of work under the guidance of Dr. D.P. Rathod at the Department of Electrical Engineering, Veermata Jijabai Technological Institute, Mumbai.

This report reflects all the work done during the entire period of Project.

Prathmesh Shinde
(18190058)

Shubh Ganatra
(181090060)

Shubham Agrawal
(181090062)

Atharva Wadekar
(181090073)

Date:

Place:

ACKNOWLEDGEMENT

We would like to thank all those people whose support and cooperation has been an invaluable asset during my Project. We would also like to thank our guide Dr. D.P.Rathod for guiding us throughout the period of my Project. It would have been impossible to complete the project without their support, valuable suggestions, criticism, encouragement and guidance.

We are also grateful to all other teaching and non-teaching staff members of the Electrical Engineering Department for directly or indirectly helping us for the completion of our project and the resources provided.

Prathmesh Shinde
(18190058)

Shubh Ganatra
(181090060)

Shubham Agrawal
(181090062)

Atharva Wadekar
(181090073)

Date:

Place:

ABSTRACT

Genre classification is an important task with many real-world applications. As the quantity of music being released on a daily basis continues to sky-rocket, especially on internet platforms such as Soundcloud and Spotify – a 2016 number suggests that tens of thousands of songs were released every month on Spotify – the need for accurate meta-data required for database management and search/storage purposes climbs in proportion. Being able to instantly classify songs in any given playlist or library by genre is an important functionality for any music streaming/purchasing service, and the capacity for statistical analysis that correct and complete labelling of music and audio provides is essentially limitless.

This study compares machine learning algorithms in their ability to automatically classify song excerpts into their musical genres. First, a review of existing techniques and approaches is carried out, both in terms of feature engineering and algorithms on offer. A dataset of music tracks from different genres is taken such that the number of samples belonging to each genre are equal. Fifty-six features are manually extracted from each sample, using audio analysis library LibROSA. Classifiers are created - a neural network, a support-vector machine, a random forest and a gradient boosting machine - and training and testing are performed on each, with the dataset created. Finally, an evaluation of the whole process is conducted, including an assessment of the choice of dataset, choice of features, and various aspects of the project management.

CERTIFICATE

This is to certify that **Prathmesh Shinde** (181090058), **Shubh Ganatra**(181090060), **Shubham Agrawal** (181090062), **Atharva Wadekar** (181090073), students of B.Tech(Electronics and Telecommunications), Veermata Jijabai Technological Institute, Mumbai have successfully completed the project titled “**Music Genre Classification using Machine Learning**” and submitted a report on the same to my satisfaction.

Dr. D.P. Rathod
Project Supervisor

HoD Electrical
Engineering Dept.

Date:

Place:

CERTIFICATE

This Project titled , “**Music Genre Classification using Machine Learning** ” by **Prathmesh Shinde** (181090058), **Shubh Ganatra**(181090060), **Shubham Agrawal**(181090062), **Atharva Wadekar**(181090073), is found to be satisfactory and is approval of the partial fulfilment of the requirement of B.Tech (Electronics and Telecommunications).

Dr. D.P. Rathod

Project Supervisor

External Examiner

Date:

Place:

Contents

1 Introduction	1
1.1 Introduction	1
1.2 Outline of all chapters	2
2 Literature review	3
2.1 Modalities	3
2.1.1 Audio	3
2.1.2 Audio Features	3
2.1.3 Mel-Frequency Cepstral Coefficients (MFCCs)	3
2.1.4 Zero Crossing Rate (ZCR)	5
2.1.5 Spectral Spread	6
2.1.6 Chroma Features	6
2.1.7 Tempo	7
2.2 Other Modalities	7
2.2.1 Textual	8
2.2.2 Visual	8
2.2.3 Multimodal	8
2.3 Final Choice of Modality	9
2.4 Classification Algorithms	9
2.4.1 Artificial Neural Network	10
2.4.2 Support Vector Machines	10
2.4.3 Random Forest	11
2.4.4 Gradient Boosting Machine	11
2. 5 Outline of chapter 3	12

3 Proposed Methodology	13
3.1 GTZAN Dataset	13
3.1.1 The Free Music Collection	14
3.1.2 Choice of Dataset	14
3.2 Data Pre-processing	15
3.2.1 Preliminaries	16
3.2.2 Feature Extraction	16
3.3 Models	17
3.3.1 Train-Test Split	18
3.3.3 Neural Network	18
3.3.4 Support-Vector Machine	19
3.3.5 Random Forest	21
3.3.6 Gradient Boosting Machine	22
3.3.7 Stochastic Gradient Descent	23
3.3.8 K-nearest neighbours (KNN) algorithm	24
3.3.9 Decision Tree	26
3.3.10 Logistic regression	27
3.3.11 Naïve Bayes	27
3.3.12 CNN	29
3.4 Web Application Implementation of the models	31
3.5 Outline of chapter 4	33
 4 Results and Analysis	 34
4.1 Discussion	34
4.1.1 Genre	34
4.1.2 Output Format	35
4.1.3 Outputs of Models Trained	35

4.1.4 Comparison between Algorithms	41
5 Conclusions and Future Scope	44
References	45

LIST OF FIGURES

Figure 2.1	MFCC Block Diagram	4
Figure 2.2	Zero Crossing rate	6
Figure 2.3	Chromagram	7
Figure 3.1	Genres in dataset	13
Figure 3.2	Extracted Features	15
Figure 3.3	Block Diagram of Data-pre processing	16
Figure 3.4	Training data flowchart	17
Figure 3.5	Architecture of Neural Network	19
Figure 3.6	Support Vector Machine	20
Figure 3.7	Random Forest Flowchart	21
Figure 3.8	Gradient Boosting	24
Figure 3.9	KNN	25
Figure 3.10	Decision Tree	26
Figure 3.11	Logistic Regression	27
Figure 3.12	Naïve Bayes Classifier	29
Figure 3.13	Convolutional Neural Network	30
Figure 3.14	Home Page	32
Figure 3.15	File Upload Page	32
Figure 3.16	Results Page	33
Figure 4.1	Confusion Matrix for Support Vector Machine	35
Figure 4.2	Confusion Matrix for Random Forest	36
Figure 4.3	Confusion Matrix for Stochastic Gradient Descent	36
Figure 4.4	Confusion Matrix for Neural Network	37
Figure 4.5	Confusion Matrix for Naïve Bayes	37

Figure 4.6	Confusion Matrix for Logistic Regression	38
Figure 4.7	Confusion Matrix for KNN	38
Figure 4.8	Confusion Matrix for Decision Tree	39
Figure 4.9	Confusion Matrix for XGBoosterandRandomForest	39
Figure 4.10	Confusion Matrix for Cross Gradient Boost	40
Figure 4.11	Confusion Matrix for CatBoost	40

Chapter 1

1.1 Introduction

The aim of this project was to compare machine learning algorithms in their ability to automatically classify song excerpts into the correct musical genre. For humans who are familiar with the genres in question, music genre recognition (MGR) is not an especially difficult task. Most people well-acquainted with Western popular music are able to identify that a given Metallica song is an example of the heavy metal genre, and that a given Status Quo song is an example of rock music, just by listening to the audio.

Importantly, they are usually able to perform this type of broad genre classification even if they are unfamiliar with the song or artist in question: ‘heard’ qualities of the sound are normally enough. Only a short sample of audio is usually needed, sometimes less than a second. A more fine-grained classification into musical subgenre usually requires some level of expertise on the part of the listener, and the greater the level of expertise, the more accurate the classification that can be made. The question is raised as to whether software can be written to perform genre classifications as well as humans, and what type of approaches work best. Despite being an interesting endeavour in and of itself, there are practical, real-world applications of automated music genre recognition, with music streaming services representing the most pertinent example.

Users may want to discover tracks that are similar to those that they are already a fan of, and it makes sense for providers to have such metadata for their songs available, in order to facilitate this kind of discovery. Given the vast and ever-growing quantity of music available on the internet, and the need for services to manage increasingly large song databases, manual analysis and annotation of individual songs does not seem like a sustainable long-term solution. From an engineering perspective, despite the cost of developing and maintaining the system, long-term expenditures could be significantly reduced with the use of accurate, automated genre-recognition systems. Large streaming services are surely already aware of the prospects of automated genre recognition techniques, but questions remain as to the best way a music genre recognition system can be architected. Various machine learning techniques have been applied to problems - including genre recognition - in the field of Music

Information Retrieval. A range of possible approaches toward the feature engineering of such projects exist. In general, machine learning classifiers will perform well when they are able to ‘understand’ what makes a data sample a member of a particular class, and, in many cases, when classes tend to be easily ‘separable’ from one another (i.e. with different classes varying significantly in terms of their average feature values).

When humans are able to recognize the genre of a given audio track, they typically do so based on ‘high-level aspects of the sound in question, e.g., the instruments in use, the feel with which they are played, the ‘overall sound’ of the song, etc. These types of emergent musical property may be difficult for software to recognize; however, given the direct availability of audio files, it is possible for software to detect aspects of audio files that humans cannot perceive through mere listening alone. Automatic genre classification based on these types of features has already been performed, with promising results. The general motivation behind such approaches is that different musical genres tend to use different types of sounds, perceptible by humans and potentially detectable by the right software as well.

This study compares several machine learning algorithms in their ability to classify songs accurately. First, a survey of existing literature is carried out, looking at how machine learning has been applied to problems of music classification and audio classification in general. Tools for extracting features from audio files are discussed, as well as techniques for classifying music based on other modalities such as song lyrics and album artwork. This is followed by a discussion of machine learning techniques fit for present purposes. Various candidate datasets are evaluated, with a justification of the final dataset chosen and an explanation of how it was pre-processed and reduced in size. An explanation of the features extracted, and an analysis of the features chosen (in terms of usefulness), are then presented.

After this, a description of the machine learning models follows, as well as a presentation of the results of each classifier. Finally, an interpretation of the results is carried out, in addition to an evaluation of design choices and an assessment of the project management. The study is concluded with the main lessons learned, and recommendations for future research.

1.2 Outline of all chapters:

We will study the modalities of the audio signal and learn how exactly different features are extracted from the audio signal and their significance in classification of the audio (Chapter 2).

We will get a basic understanding of the fundamentals of different models and see their application in training and predicting the genre of the dataset that will be provided during testing (Chapter 3).

Accuracy of each model is calculated against a testing dataset and the most accurate model is considered for prediction of music genre, furthermore the results are obtained by such analysis (chapter 4).

We finally conclude the report by discussing the future aspect of the project using proper references.

Chapter 2

Literature Review

2.1 Modalities

2.1.1 Audio

Machine learning typically requires each data example to be represented numerically, often as a vector of relevant information or ‘feature vector’. To classify based on audio alone, audio files are often processed such that the relevant characteristics are extracted and stored in vector format. The rest of this section details the types of characteristics that can be derived from audio files for use in machine learning, and examples of studies that have utilized these techniques for classification purposes.

2.1.2 Audio Features

There are three main types of audio feature employed for music classification: timbral texture features, rhythmic features, and pitch content features. Detailing every possible candidate feature is outside the scope of this project; this section will instead outline a selection, which are among the most commonly used for classification (some of which were used in this project):

1. MFCCs (timbral texture feature)
2. Zero Crossing Rate (timbral texture feature)
3. Spectral Spread (timbral texture feature)
4. Chroma Features (pitch content feature)
5. Average Tempo (rhythmic feature)

2.1.3 Mel-Frequency Cepstral Coefficients (MFCCs) (timbral texture feature)

MFCCs are one of the most common feature types used in audio classification of all kinds, having been used in speech and ambient noise recognition as well as music genre classification. Unlike the estimated tempo of a given audio signal, which consists of a single value, ordinarily between 13 and 30 MFCCs are extracted, with the exact number of coefficients a decision made by the researcher. In this project, of the 56 features extracted from the audio file dataset, 40 were MFCCs consisting of their means and variances; thus, it is worth examining the feature type in some detail.

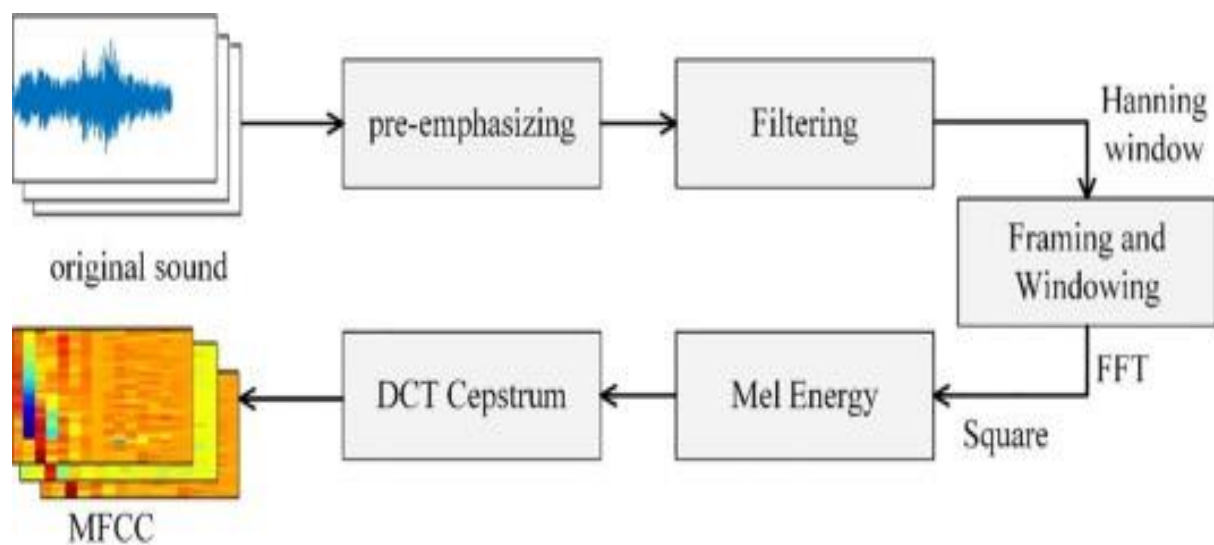


Fig 2.1

MFCCs can be better understood by understanding the process by which they are derived from an audio file, which ordinarily runs as follows:

- An audio file is divided into small frames (a process known as 'windowing'), each lasting 20-40ms. The resulting vector is a time series that represents the overall audio file.
- For each frame, an estimation of the power spectrum is calculated, by use of a Fourier Transform. The power spectrum of an audio frame is roughly equivalent to the power present for each of the frequency bands within that frame.
- A 'Mel-spaced Filterbank' is used on each of the output sets from step 2. The Filterbank is spaced according to the mel scale, to give emphasis to the frequency bands that are most relevant in human perception (humans find it easier to perceive differences in the lower registers, i.e., the bassier

frequencies, so the filters are more narrowly spaced in these lower registers than in the higher registers).

- Each of the filters corresponds to a particular frequency band, and removes all of the values that fall outside this range. The result is a series of 'filterbank energies'; one value per filter.
- After computing the filterbank energies, their logarithms are calculated.
- A Discrete Cosine Transform is performed on the logarithms, and approximately half of the resulting values are dropped. The result is a time-series vector of MFCCs: one set of MFCCs for each frame of the audio file.

It is further possible to find the mean values for each MFCC of an audio file. For example, to find a signal's mean MFCC5 value, the values of MFCC5 across all windows of the signal would be added up, and this value would then be divided by the number of windows in the signal. This process can be applied for each MFCC to find the mean MFCC values.

MFCCs are used in music genre classification to detect timbre , which can be defined as the 'quality' or 'colour' of a sound. Different genres of music use different sets of instruments, resulting in striking timbral differences - compare the soft sound of Chopin's piano compositions with the harshness of punk rock.

Even when two genres use the same set of instruments, they are often used to totally sonic different effect; e.g. Electronic Dance Music often features a bass-heavy and punchy kick drum, in comparison to jazz music, which typically uses the kick drum in a much more subdued way. The timbral qualities of a song are detectable in its spectral information, and use of MFCCs has proven to be a powerful way of representing this content for use in machine learning.

2.1.4 Zero Crossing Rate (timbral texture feature)

A second feature commonly used in music genre recognition is the zero-crossing rate (ZCR). The ZCR of a signal is defined as the rate at which a signal changes from positive to negative or negative to positive. In the context of audio, this refers to the number of times the amplitude of the signal passes through a value of zero, within a given time interval. It is commonly used as a measure of the smoothness of an audio signal, and has been used as a feature in various forms of audio classification. For example, it has been used to separate speech recordings into periods of 'voiced' speech (sounds made when vowels are

spoken) and ‘unvoiced’ speech (sounds made when consonants are spoken). In the context of music information retrieval, ZCR has been used to identify the presence of percussive sounds; a factor relevant to genre recognition because percussive sounds are utilized by some genres more than others. Like MFCCs, ZCR is another feature that can be used to understand the timbre of an audio file.

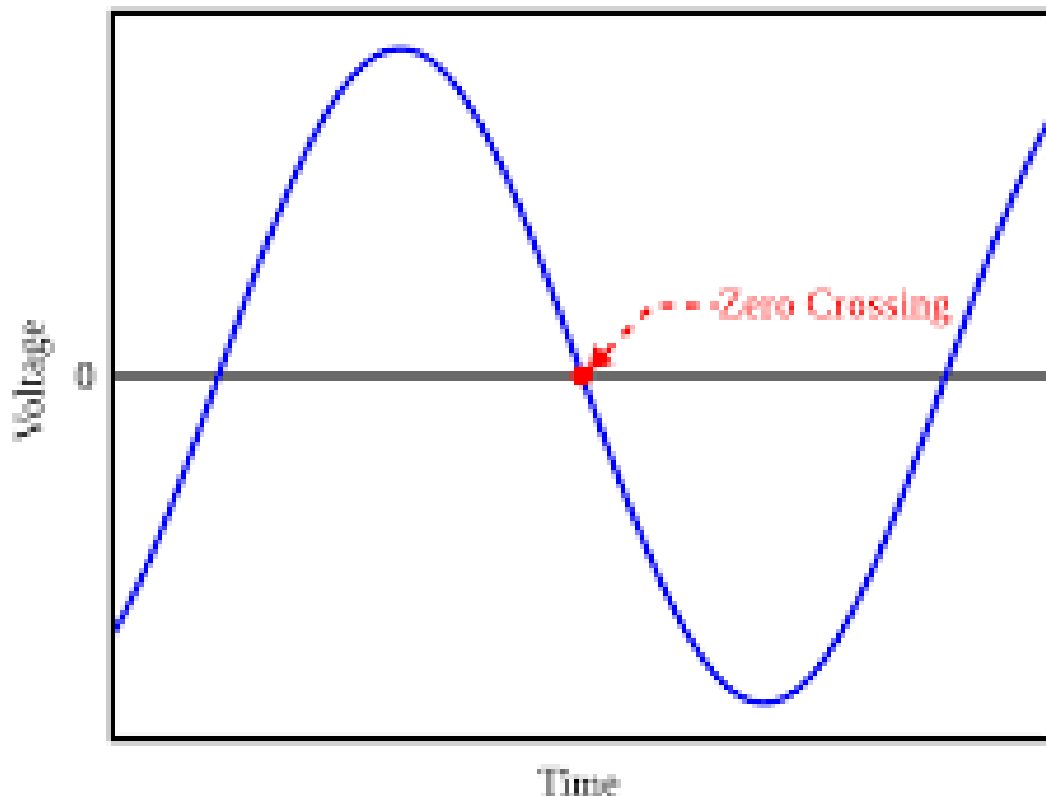


Fig 2.2

2.1.5 Spectral Spread (timbral texture feature)

‘Spectral spread’ is understood in the context of ‘spectral centroid’, which refers to the ‘centre of gravity’ of the spectrum of a signal. Perceptually, spectral centroid relates to the brightness of a signal’s sound. The spectral spread of a signal describes the ‘average deviation of the rate map around its centroid’ ; i.e., the variance.

Noisier sounds imply a higher spectral spread, whereas signals with a spectrum ‘tightly concentrated’ around the centroid will have a lower spectral spread. Like

MFCCs and ZCR, spectral spread is used to understand the timbre of an audio file, and is another feature that has been used in music genre recognition.

For example, it is seen that spectrograms of electronic music excerpts are typically spread more widely around their centroid than are excerpts of jazz.

2.1.6 Chroma Features (pitch content feature)

Chroma-based features concern the musical notes that are played throughout a song. Various techniques for detecting musical pitch exist, with newer approaches focusing on pitch detection of polyphonic sequences. Audio analysis software packages are able to estimate the intensity with which specific notes are present within an audio file, as well as how these changes take place over time.

The chromagram in figure 2.1 (Meinard.mueller, 2016) represents a C-major scale played over the course of several seconds.

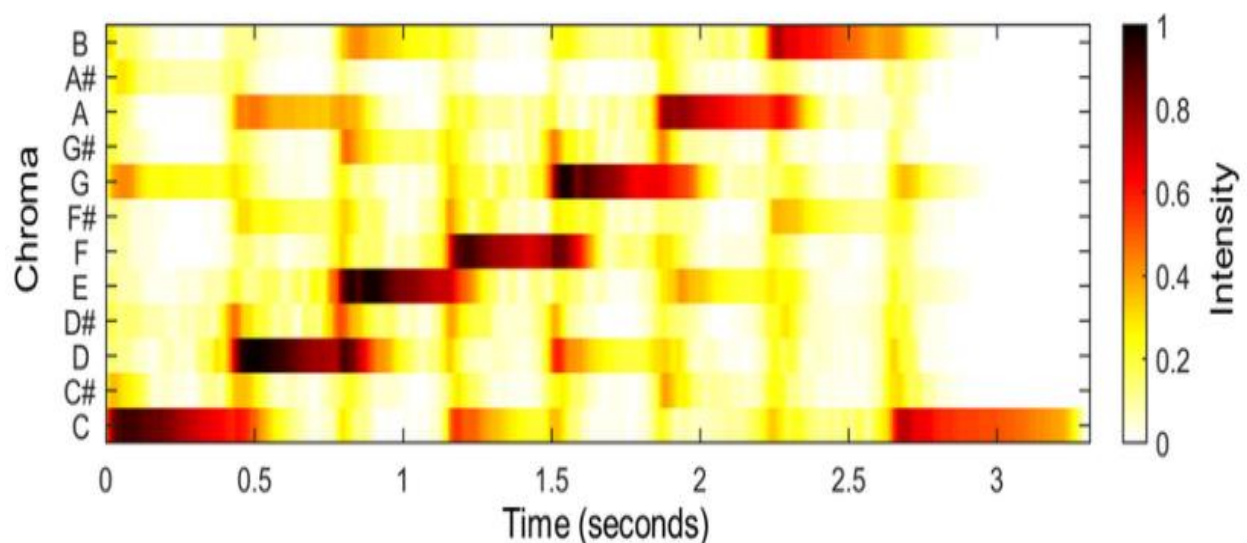


Fig 2.3

Information on the notes played most frequently throughout a song may be useful in classification for a number of reasons - for example, because different genres of music may tend towards different key signatures. Typically, the most frequently played note of a song will be its root note; i.e., the note of its key signature (e.g. 'A' is the root of A major).

Guitar-based music is more likely to be played in E or A than F or A#, because E and A chords (and the chords of these key signatures) are easier to play on standard-tuned guitars than are F and A# chords. Other styles of music may have no special leaning towards these particular keys, or may lean toward other key signatures entirely.

Therefore, all else being equal, if a song is detected to be in E or A, we might have additional reason to believe that it is in a guitar-based style, unlike if it is detected to be in F or A#.

2.1.7 Tempo (rhythmic feature)

Finally, an estimation of musical tempo is sometimes used as a feature for classification. Although a song may include several tempo changes throughout its duration, most popular music is set to a fixed number of beats per minute.

Given the rhythmic emphasis placed on each beat of a bar (e.g. a typical 4/4 drumbeat will involve a high-hat played on each beat, with a snare drum on the 2nd and 4th of each bar), it is relatively straightforward to estimate the number of beats per minute, as long as these rhythmic emphases can be detected.

Tempo may act as a suitable feature for genre classification because different genres may typically be played at different speeds.

Within certain subgenres, this is trivially true (e.g. speed metal, as the name suggests, is played at high tempos, whereas doom metal tends to be played much more slowly); it is plausible that this applies to broader genre categories as well.

2.2 Other Modalities

Analysis of audio features does not represent the only avenue for automatic music genre classification; other modalities such as text and imagery have been used as well.

This section will explore these alternate approaches and justify the final choice of modality used in this project.

2.2.1 Textual

Songs and albums often have several text sources associated with them. These include primary sources, such as song titles and lyrics, and secondary resources, such as album reviews. Lyrically, different genres clearly tend to use different types of words.

For example, rap lyrics are more likely to include topical words such as 'game' and 'rhyme' than metal lyrics, which tend to favour darker terms such as 'death' and 'shadow'. Emergent properties such as rhyme scheme and song structure have also been used as basis for classification.

Fell and Sporleder used an n-gram model to classify lyrics into different genres, with some success, although some genres (e.g. rap) were more easily detectable than others (e.g. folk was often confused with blues or country).

Mayer, Neumayer and Rauber used a range of different feature combinations, and categorized lyrics with several classifiers, including k-Nearest Neighbours and Naïve Bayes.

2.2.2 Visual

There has been comparatively little work done on classification of music based on its album artwork.

Automatic classification of images by genre has been attempted for other art forms, with some success.

While investigating whether paintings could be automatically classified into their art genres, Colour and texture-based features were used as input to a range of classifiers including a neural network and a decision table and the results obtained were promising.

Album covers of disparate genres tend to differ in content and style (e.g. violent imagery in death metal album covers vs. its relative absence in k-pop artwork).

2.2.3 Multimodal

The prospect of multimodal music classification - classification based on several modalities used in conjunction with one another - has received relatively little attention over the years, although interest is growing. In their 2018 paper, by Oramas et. al (2018) details two experiments.

The first employs separate neural networks to classify albums into distinct genres: first using audio features (of the songs), then using visual features (of the album art), and finally according to a 'multimodal feature space' made by fusing audio and visual representations.

Their findings suggest that combining the representations yielded superior results to using either modality by itself. Their second experiment introduced text representations learned from album reviews.

Neural networks were trained according to each modality, as well as for different modality combinations, with the most accurate approach taking all three modalities into account.

It is seen when using visual features only, performance of the classifiers was still noticeably lower than that achieved using audio features such as Temporal Variants.

The addition of visual features did appear to slightly increase the accuracies of the classifiers in some cases, but we found that this improvement varied with the type of classifier used and also some accuracies decreased when both sets of features were considered for some models.

Thus, as regards to automated music video classification, analysis of video content may represent an unnecessary addition to the process, if the audio for the music videos is readily available to be used and analysed.

2.3 Choice of final Modality

For this study, only audio features were used. There were several motivations behind this choice. First, after consideration of the time and resources available, it was determined early on that multimodal classification was outside the scope of this project.

Selecting one modality allowed for deeper, more thorough exploration of that particular modality, and probably yielded better results than would have been

achieved through a multimodal approach. Secondly, text-based approaches typically make use of natural language processing techniques that were also determined to be outside the scope of this project's aims.

Although genre classification of music via album artwork alone represents an interesting and novel possibility, it is not immediately clear that these types of classifier perform especially well, and there is relatively little existing research in this area for the understanding of its working.

Finally, as discussed above, the majority of existing literature and resources in this area concern music genre classification via audio features. There also exist a number of useful and relevant audio analysis libraries which could be made use of in the process of feature extraction. Therefore, after these assessments had been made, an audio-based approach was chosen.

2.4 Classification Algorithms

The remainder of this chapter will look more closely at four machine learning algorithms that are frequently used for classification problems, and will include discussion of how such methods can be applied to the task of music genre recognition.

2.4.1 Artificial Neural Network

Artificial neural networks are a popular form of machine learning classifier, consisting of layers of connected nodes that transform input data to some kind of output (dependent upon the specific implementation).

In the context of classification, neural networks learn models from training data, in order to predict the class of unseen data samples. Various kinds of neural network exist, including fully-connected, convolutional, and recurrent, and networks can be structured with varying numbers of nodes and layers according to design choices made by the engineer.

A number of strategies exist for combating overfitting and reducing model complexity, and, when architected correctly, neural networks have proven to be especially effective for classification problems.

The exact structure of the network used for this project will be provided in the 'models' section of this report.

A range of approaches can be taken toward neural network input. The shape of the first layer of the network will depend upon choices made in the feature engineering stage of the project. For example, in image recognition, the number of 'input node' will usually equal the number of pixels of each image of the dataset. Each pixel is represented by a number, and, if structured correctly, the network can be trained to learn 'features' of the images, e.g. the curve of a '2' in handwriting recognition.

However, when features have been manually selected from the data, as is the case for this project, the number of input nodes is determined by the number of features extracted in this way.

Audio data can be described as 'time-series data', in that it describes sonic events that take place over time. Because of this, there exists the possibility of representing audio files as time-series vectors, in terms of features such as MFCCs.

Recurrent Neural Networks make use of loops and are able to 'remember' information between stages of the network's processing. This makes them especially suited to sequential and time-based data.

2.4.2 Support-Vector Machine

SVMs are another type of supervised learning machine learning algorithm. In classification problems, data points of the same class will oftentimes cluster together on graphs that plot their feature values.

In order to make predictions for new data points, a 'n-1 dimensional hyperplane' can be used to carve up the graph, where n = the number of features. A new data point falling on one side of the hyperplane will be predicted to be of one class; a data point falling to the other side will be predicted another.

SVMs attempt to find the optimal hyperplane for a given dataset, ordinarily defined as the hyperplane that maximizes the margin between the classes.

SVMs can be initialized with a range of different kernel functions, which create different types of data divisors. A linear kernel function attempts to divide data points with a linear hyperplane, whereas a 'kernel trick' can be used to create

polynomial decision boundaries, by effectively mapping the data points into higher dimensional space so that a dividing hyperplane can be drawn.

Depending on how disparate two given classes are, the clusters may be 'far apart' on the graph, or there may be some degree of overlap, and the success of a support vector machine implementation will partly depend on the ease with which the classes can be separated.

If, as has been argued above, different musical genres produce distinct average feature values, then support vector machines may prove promising for music genre classification, as the clusters of each genre could be separated with hyperplane divisors relatively easily.

2.4.3 Random Forest

Random forest is a form of ensemble learning, meaning that it combines models with the aim of producing a model that is better than any of the individual models by itself. In applying the random forest technique to classification problems, a data point is put through a number of decision trees, and the predicted class of the data point is equal to the mode average of the outputs of the individual trees considered together.

In order to predict the class of a sample, a decision tree passes the values of the features through a series of conditional statements. This continues until a 'leaf node' - which designates the class predicted - has been reached. In random forests, an element of randomness is added through 'feature bagging'.

Various hyperparameters can be tuned, such as the number of decision trees in the forest and the maximum number of features used per tree. As such, random forest was deemed suitable to be one of the classification algorithms for this project.

2.4.4 Gradient Boosting Machine

Like random forest, boosting is another ensemble technique. Gradient boosting machines use decision trees in the context of an additive model, in which gradient descent is used to minimize the level of loss as more trees are added.

Gradient boosting has proven to be among the most effective machine learning algorithms, often outclassing the more well-known alternatives.

XGBoost was used to implement gradient boosting for this project. Hyperparameters include the minimum child weight, gamma, and the maximum depth of the trees; tuning of such hyperparameters will be discussed later in this report.

2. 5 Outline of chapter 3:

In the next chapter we will dive into the selection of dataset, pre-processing of the dataset, training of dataset using 12 different models with their brief description. Finally, we show our results on our website on the result page.

Chapter 3

Method

This chapter details the methods used to select a dataset, pre-process the data, extract and analyse features, and create and test the machine learning models. Python 3 was used for the development of this project, due to the extensive machine learning libraries and online resources available for it.

3.1 GTZAN Dataset

The GTZAN dataset, consists of 10 different groups of 100 files of 30-second song excerpts, with each group representing one genre, totalling 1000 audio files. It is a relatively well-known resource in the field of Music Information Retrieval (MIR).

The dataset comes pre-organised into folders - one per genre - making it fairly straightforward to navigate, and the balanced nature of the dataset makes it appealing for machine learning projects.

However, despite these attractive features, GTZAN bears a number of disadvantages. First, with only 100 items per class, it is a relatively small dataset, in the context of other machine learning projects.

Deep learning is often considered to thrive on large amounts of data and having fewer training examples means that the classifiers have less information from which to build their models. It therefore made sense to see whether a larger dataset would be available.

Genre	Number of tracks
Blues	100
Classical	100
Country	100
Disco	100
Hip-Hop	100
Jazz	100
Metal	100
Pop	100
Reggae	100
Rock	100
Total	1000

Fig 3.1

3.1.1 The Free Music Collection

The size of this dataset makes it ideal for classification purposes, and the fact that the audio files are available to download means that features from the audio can be extracted directly. The final dataset downloaded for this study was the 'large' version, which consists of 30s samples of all 1000 songs.

30-second samples are usually enough for classification purposes, and the use of smaller song excerpts makes the dataset much easier to acquire than if full tracks had to be downloaded.

The large dataset is unbalanced - i.e., different genres were represented to different extents - but, as will be detailed in the next section, two smaller versions of this dataset were created, both genre-balanced, for purposes of classification.

3.1.2 Choice of Dataset

For the final dataset we took 1000 audio samples such that each genre had 100 audio samples of 30 seconds assigned to it. In order to increase the size of the dataset we divided each audio sample into 10 audio samples of 3 seconds each.

The resulting dataset now has 1000 audio samples for each genre. The dataset has 10 genres namely, Blues, Classical, Country, Disco, Hip-hop, Jazz, Metal, Pop, Reggae, Rock.

mfcc1_mean	mfcc8_mean	mfcc15_var	rms_var		
mfcc1_var	mfcc8_var	mfcc16_mean	spectral_centroid_mean		
mfcc2_mean	mfcc9_mean	mfcc16_var	spectral_centroid_var		
mfcc2_var	mfcc9_var	mfcc17_mean	spectral_bandwidth_mean		
mfcc3_mean	mfcc10_mean	mfcc17_var	spectral_bandwidth_var		
mfcc3_var	mfcc10_var	mfcc18_mean	spectral_rolloff_mean		
mfcc4_mean	mfcc11_mean	mfcc18_var	spectral_rolloff_var		
mfcc4_var	mfcc11_var	mfcc19_mean	zero_crossing_rate_mean		
mfcc5_mean	mfcc12_mean	mfcc19_var	zero_crossing_rate_var		
mfcc5_var	mfcc12_var	mfcc20_mean	harmony_mean		
mfcc6_mean	mfcc13_mean	mfcc20_var	harmony_var		
mfcc6_var	mfcc14_mean	chroma_stft_mean	tempo		
mfcc7_mean	mfcc14_var	chroma_stft_var			
mfcc7_var	mfcc15_mean	rms_mean			

Fig 3.2

Features such as Mel-Frequency Cepstral Coefficients, Zero Crossing rate, Chroma Features, Spectral Spread and Tempo were extracted from each audio sample. The feature vector had 56 features. All the considered models were trained using this dataset.

3.2 Data Pre-processing

Each audio track in the samples is divided into 10 audio tracks of 3 seconds each. Database of the complete collection was created and stored in a .csv file.

Feature Vector Extraction is done using the libROSA package in python. libROSA is a python package for music and audio analysis which provides the building blocks necessary to create music information retrieval systems.

Each audio file is taken and from that, its feature vector is extracted. The features extracted Mel-Frequency Cepstral Coefficients, Zero Crossing rate, Chroma Features, Spectral Spread and Tempo.

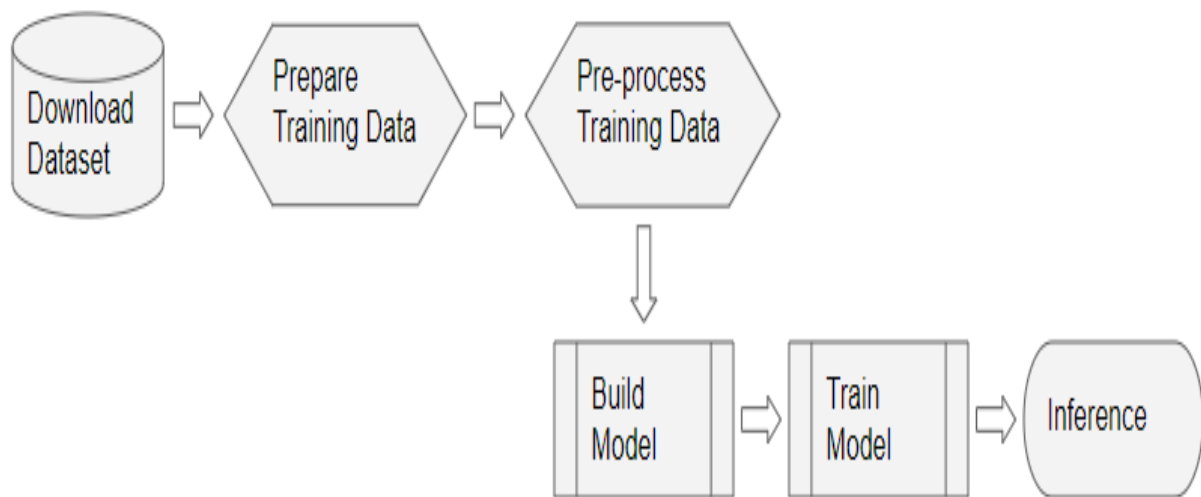


Fig 3.3

3.2.1 Preliminaries

Several stages of pre-processing were required in order to ready the dataset for classification. First, 'features.csv' was downloaded from the FMA webpage. This provided information on the genres associated with each song. We also divide the collection of 1000 dataset into 3 seconds so that the model can be trained with a wider range of dataset.

Several genres were removed from the dataset. 'Spoken' tracks were deleted because spoken-word poetry does not really constitute music, at least with regard to the particular project at hand. 'Experimental' tracks were deleted

because experimental music is broad and diverse, and songs under the label of ‘experimental’ may bear little sonic similarity to one another, making the genre hard to define and potentially making the tracks very difficult to classify. ‘international’ music was discounted because again, there may be too much diversity under such a broad label.

We have modified the dataset in order to generate equal number of training samples of all the 10 genres so that the model is not biased with overfitting and underfitting of the dataset is avoided.

3.2.2 Feature Extraction

In order to represent the tracks numerically, 56 audio features were extracted from each track. These values represent the average over the entire track (i.e., a track’s MFCC1 value was defined as the average value of MFCC1 across all windows of that track).

The first 41 were extracted using LibROSA audio analysis library (librosa.github.io/librosa); the final 13 were extracted using Aubio (aubio.org), which required conversion from mp3 to WAV format first.

In total, this consisted of: 40 MFCCs which has mean and variance values, spectral centroid, spectral bandwidth, spectral contrast, spectral flatness, spectral rolloff, chroma stft, chroma cqt, chroma cens, tempo, zero crossing rate, root mean square, energy, high-frequency content, spectral complex, spectral phase, spectral difference, Kullback-Liebler, modified Kullback-Liebler, spectral flux, spectral slope, spectral spread, spectral skewness, spectral kurtosis, and spectral decrease.

Several of these features were discussed earlier in this paper in terms of their usefulness in music genre recognition; in general, the features chosen were selected because they represent sonic information relevant to this type of classification. ‘Tempo’ was the only rhythmic feature extracted, and ‘chroma cqt’, ‘chroma_stft’ and ‘chroma_cens’ were the only pitch-content features extracted; the rest all related to the timbre and texture of the tracks.

All of the audio features used are described in their documentation as ‘spectral features’. Feature scaling was used in the creation of the following two datasets, in order that each feature played an equivalent role when put through the

machine learning classifiers. However, not each feature was equally informative, as shall be discussed in the analysis section of this report.

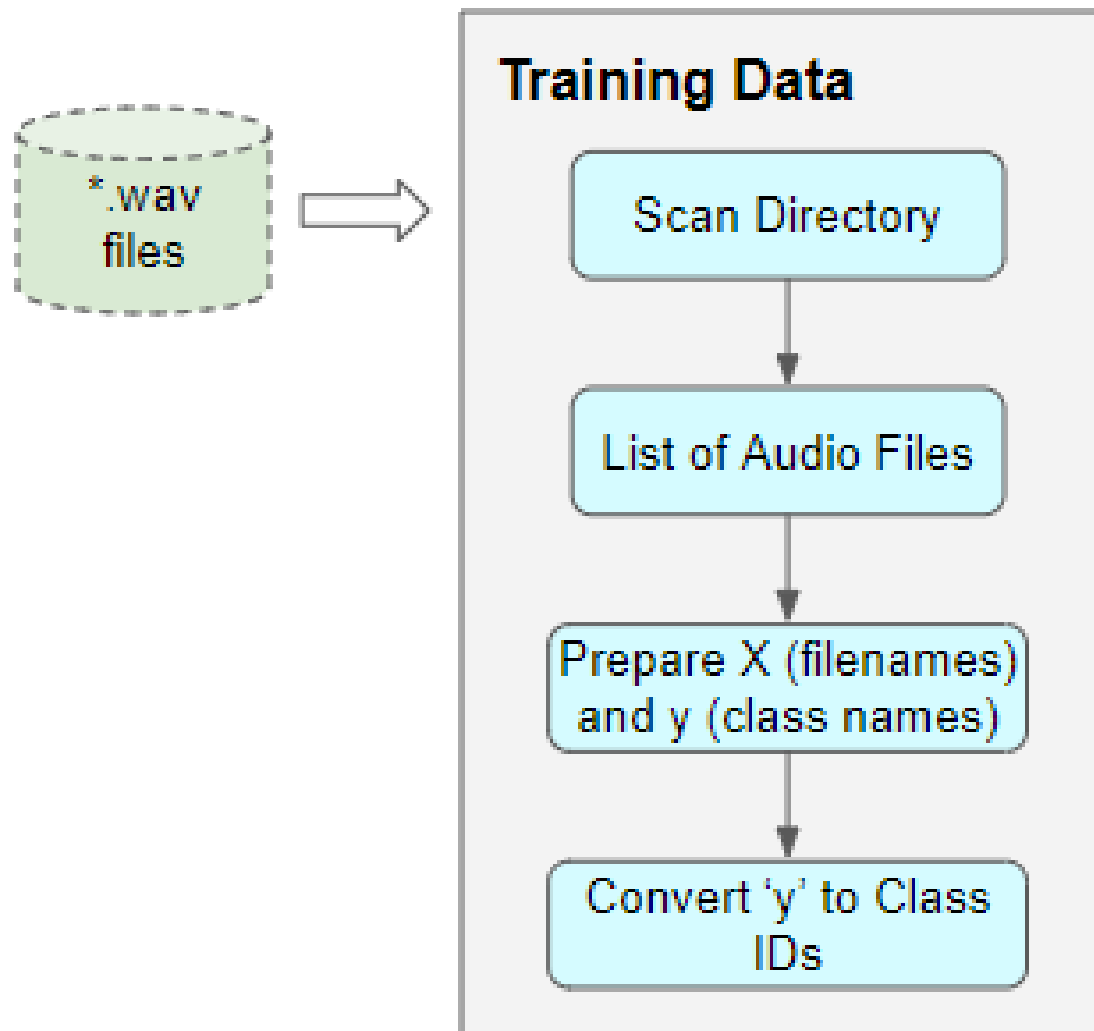


Fig 3.4

3.3 Models

This section will first detail the cross-validation procedure used to train and validate each machine learning model.

It will then detail how each of the four classifiers was implemented, and how hyperparameters were selected, as well as details of the training process.

3.3.1 Train-Test Split

A test size of 10% was used in all experiments. When using the standard dataset, the split was performed using Scikit-learn's train-test split function. For the augmented dataset, the train set and test set had to be imported manually and separately, as they had been exported separately as different csvs during the earlier SMOTE process.

3.3.2 Neural Network

The neural network was the first classifier to be implemented. A Sequential Keras model was used because it is a straightforward model that is relatively easy to tune, and this allowed more time to be spent tuning the other classifiers.

3.3.3 Structure of the Network

The input layer to the network was given the same number of nodes as features of the data (fifty-four). The output layer, which was activated with Softmax, was given five units, one for each genre.

Several designs for the internal structure of the network were tested. In general, the more layers and units used, the more a network is able to learn complex features from the data.

However, care must be taken not to overfit the training data; hence, regularisation was used to prevent this from occurring (see below). More complex architectures were initially experimented with, but the best result was achieved with a five-layer network in which the number of nodes decreased as the layers went on, as is common for neural networks of this kind.

The three hidden Dense layers were, like the input layer, activated with Relu. Relu is considered a good all-round activation function for a variety of reasons, e.g. it has been shown to be faster than alternatives like tanh in many cases.

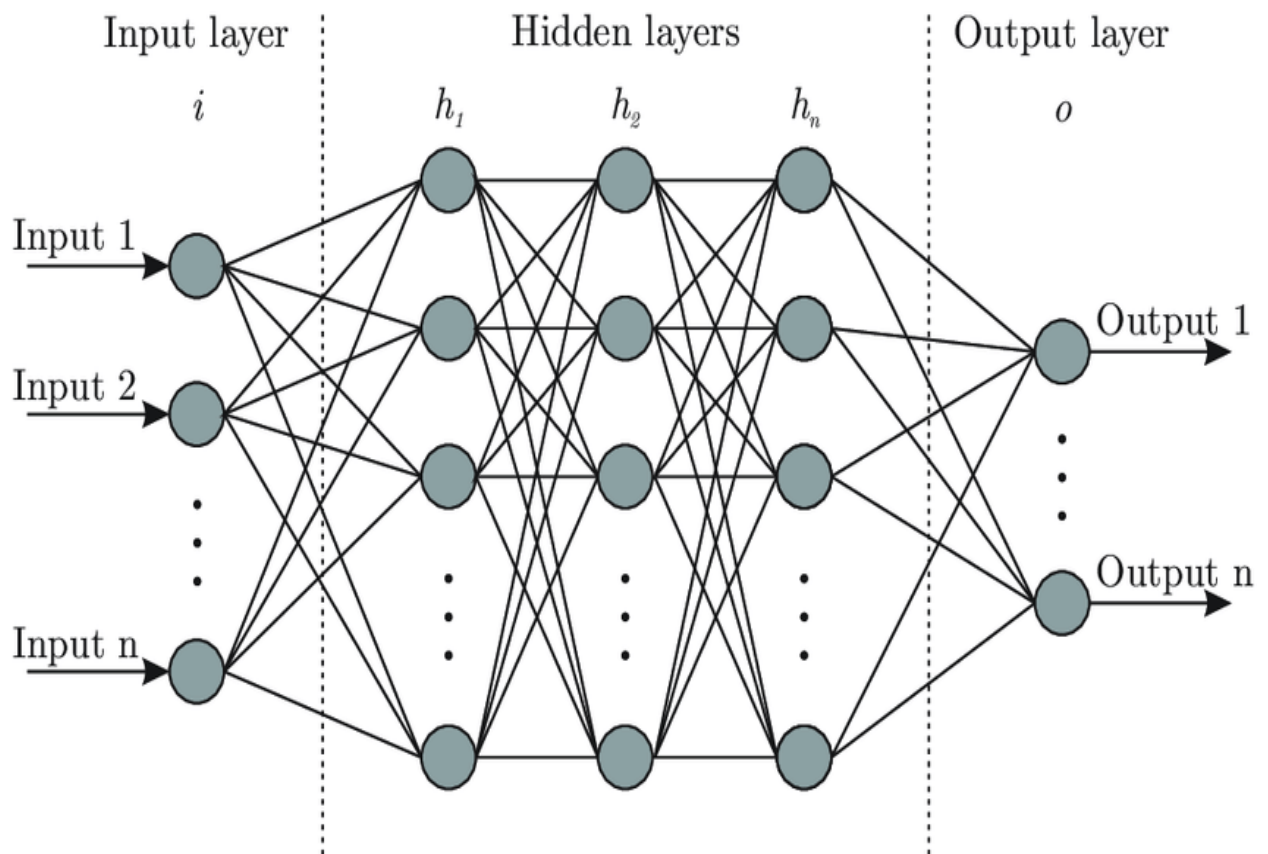


Fig 3.5

Optimizer

Adam was used as the optimizer because, like Relu, it has been shown to perform well in a wide variety of systems .

Regularisation

Dropout layers were added in between the layers of the network for regularisation purposes. Several values were tested; it was revealed that a value of 0.2 had the best effect on reducing overfitting without causing underfitting.

Training & Testing

A number of different epochs and batch sizes were experimented with. Grid search was performed to help find the best values. Testing was performed by comparing the model's predicted labels for the test data with the data's true labels, and returning the accuracy score as a percentage.

3.3.4 Support-Vector Machine

The support-vector machine was implemented with SVC from `sklearn.svm`. Grid search is a time-consuming technique, meaning not every possible parameter could be tested; therefore only a select few were used in the tuning process (this approach was also adopted for the random forest and gradient boosting machine).

Kernel Type

The kernel type determines the way in which the decision boundary of a SVM is drawn. Linear kernels are used when the data is considered to be linear separable, but given the high-dimensional nature of the data in question, this was deemed to be unlikely. Several kernel types were experimented with; initial testing revealed that the 'rbf' (radial basis function) kernel proved the most successful, and this was selected for the final model.

C

'C' is another relevant hyperparameter, used to determine the level of regularization and model complexity, by specifying the degree to which misclassified points should negatively impact the score of a model. Several values of C were tested.

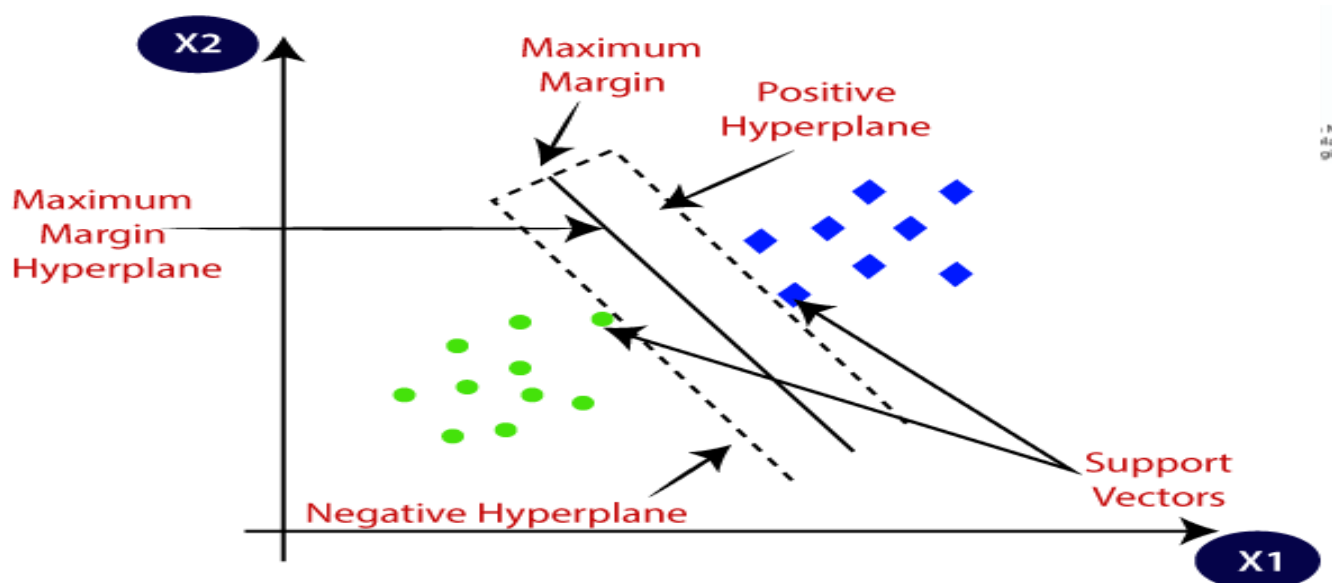


Fig 3.6

Gamma

'Gamma' is used to determine the length of the influence of a single training example in the calculation of the dividing hyperplane. Several values were tested.

Shrinking

When the shrinking hyperparameter is set to 'True', the shrinking technique 'tries to identify and remove some bounded elements, so a smaller optimization problem is solved'. Testing revealed that setting the shrinking hyperparameter to 'True' yielded better results.

Training & Testing

Grid search was again used to help identify the best hyperparameters, and after a number of tests, the following hyperparameters were chosen: kernel=rbf, C=1, gamma= 0.1, shrinking=True. These proved to be the best found for both the standard dataset experiments and the augmented dataset experiments

3.3.5 Random Forest

The RandomForestClassifier from sklearn.ensemble was used for the random forest, because, like sklearn's SVC, it is relatively straightforward to implement and tune.

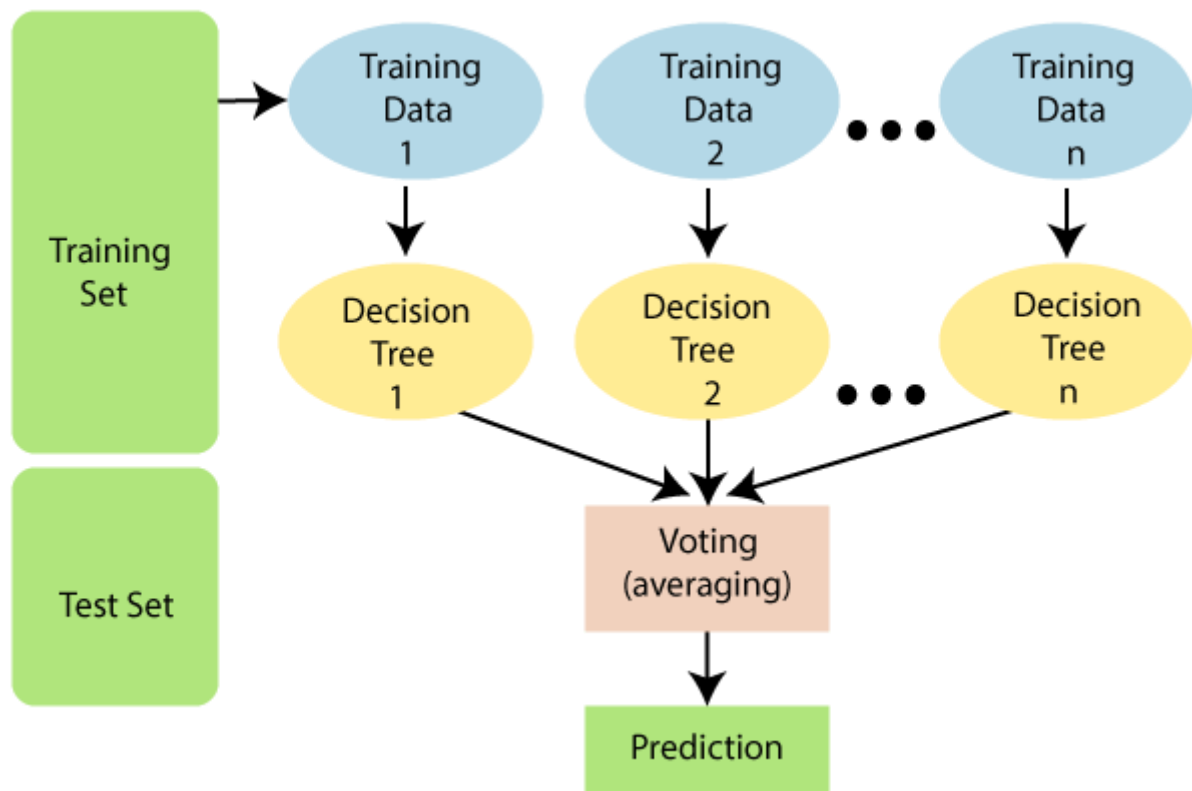


Fig 3.7

Number of Estimators

One of the primary hyperparameters to set is the number of decision trees (also known as estimators) in the forest. More trees usually leads to better accuracy; the main downside being increased running time of the algorithm. A range of values were tested.

Criterion

This hyperparameter is used to 'measure the quality of a split' during the creation of decision trees in the forest. Sklearn's implementation supports Gini Impurity and Information Gain. Both values were tested during the grid search process.

Max Features

Another important hyperparameter is the maximum number of features that is used per tree. By default, there is ordinarily no limit on this number. Several values were tried in the initial testing of the algorithm; it was found that the 'automatic' (i.e. default) value yielded the best result.

Max Depth

Increasing the maximum depth of each tree in the forest may allow for a more detailed learning process. Choosing to place no limits on the depth of the trees can improve accuracy, although this does increase model complexity and thus the likelihood of overfitting. This hyperparameter was tested during the grid search process.

Training & Testing

After grid search was used to identify the best hyperparameters, the following were chosen: `n_estimators=1000`, `criterion=gini`, `max_features=auto`, `max_depth=10`. These applied for both the standard and augmented datasets.

3.3.6 Gradient Boosting Machine

Min Child Weight

The min child weight refers to the 'Minimum sum of instance weight needed in a child'. Higher values of 'min child weight' will result in models that are less complex, so as a parameter it can be used to prevent overfitting. Several values were tested.

Learning Rate

A faster learning rate will result in shorter training times, but means that the model is more likely to overfit the data. A number of values between 0 and 1 were experimented with.

Max Depth

The 'max depth' hyperparameter refers to the maximum tree depth for the base learners. Again, increasing this value leads to greater model complexity, which can be advantageous but can lead to overfitting. Several values were tested.

3.3.7 Stochastic Gradient Descent

Gradient descent is an iterative first-order optimization algorithm used to find a local minimum/maximum of a given function. Gradient Descent is an optimization algorithm used for minimizing the cost function in various machine

learning algorithms. It is basically used for updating the parameters of the learning model.

Stochastic gradient descent This is a type of gradient descent which processes 1 training example per iteration. Hence, the parameters are being updated even after one iteration in which only a single example has been processed. Hence this is quite faster than batch gradient descent. But again, when the number of training examples is large, even then it processes only one example which can be additional overhead for the system as the number of iterations will be quite large

In mathematical terminology, Optimization algorithm refers to the task of minimizing/maximizing an objective function $f(x)$ parameterized by x . Similarly, in machine learning, optimization is the task of minimizing the cost function parameterized by the model's parameters.

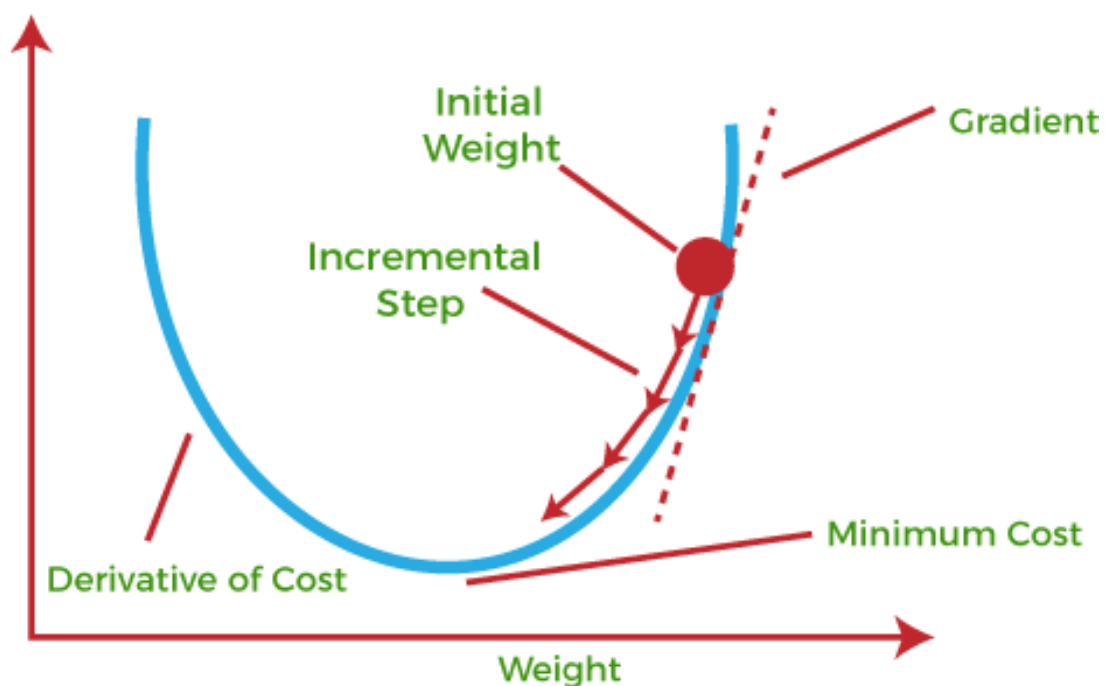


Fig 3.8

The best way to define the local minimum or local maximum of a function using gradient descent is as follows:

If we move towards a negative gradient or away from the gradient of the function at the current point, it will give the **local minimum** of that function.

Whenever we move towards a positive gradient or towards the gradient of the function at the current point, we will get the **local maximum** of that function.

3.3.8 K-nearest neighbours (KNN) algorithm

It is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well –

Lazy learning algorithm – KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.

Non-parametric learning algorithm – KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

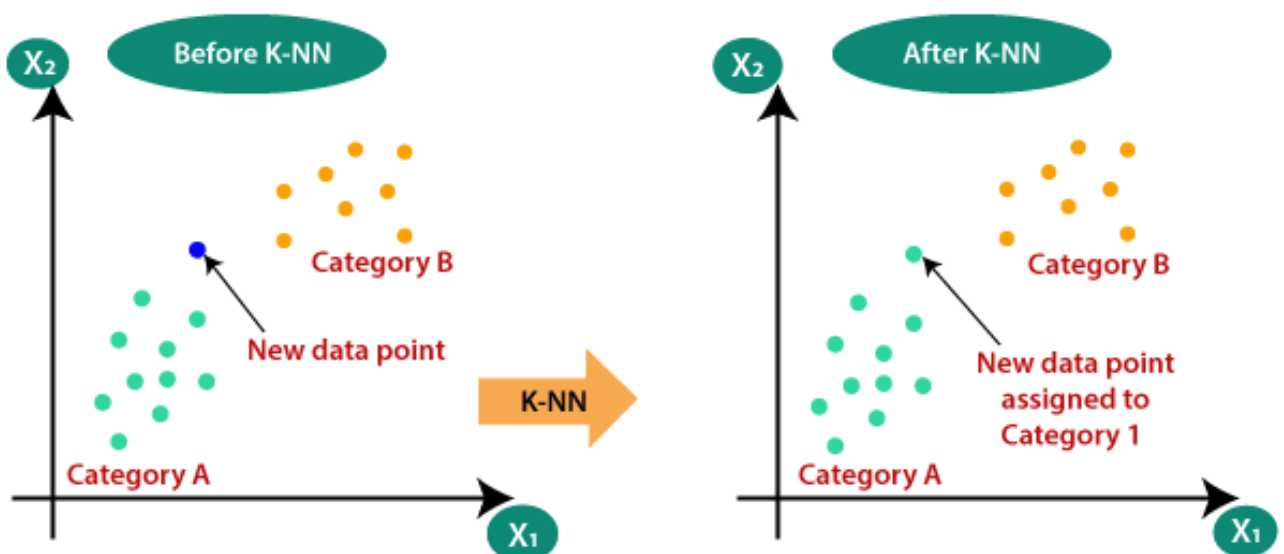


Fig 3.9

Working of KNN Algorithm

K-nearest neighbours (KNN) algorithm uses 'feature similarity' to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps –

Step 1 – For implementing any algorithm, we need dataset. So, during the first step of KNN, we must load the training as well as test data.

Step 2 – Next, we need to choose the value of K i.e., the nearest data points. K can be any integer.

Step 3 – For each point in the test data do the following –

3.1 – Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.

3.2 – Now, based on the distance value, sort them in ascending order.

3.3 – Next, it will choose the top K rows from the sorted array.

3.4 – Now, it will assign a class to the test point based on most frequent class of these rows.

3.3.9 Decision Tree

Is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

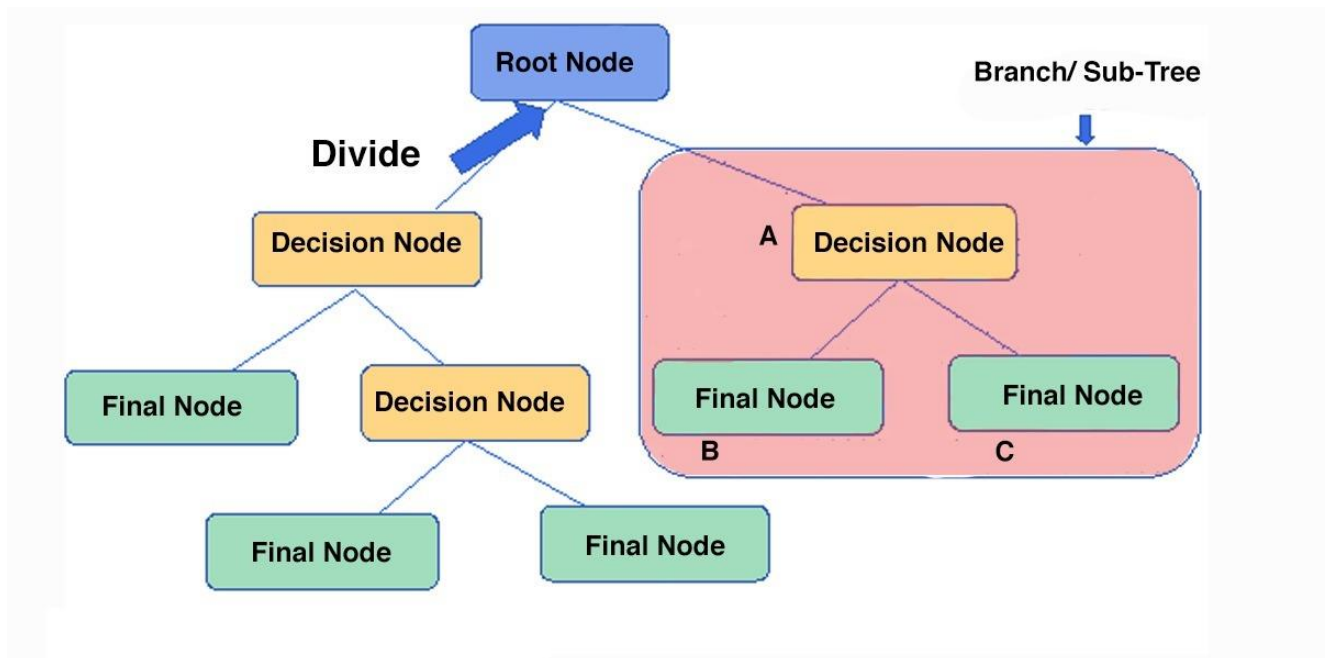


Fig 3.10

The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

3.3.10 Logistic regression

Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable(or output), y , can take only discrete values for a given set of features(or inputs), X .

Contrary to popular belief, logistic regression IS a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”. Just like Linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function.

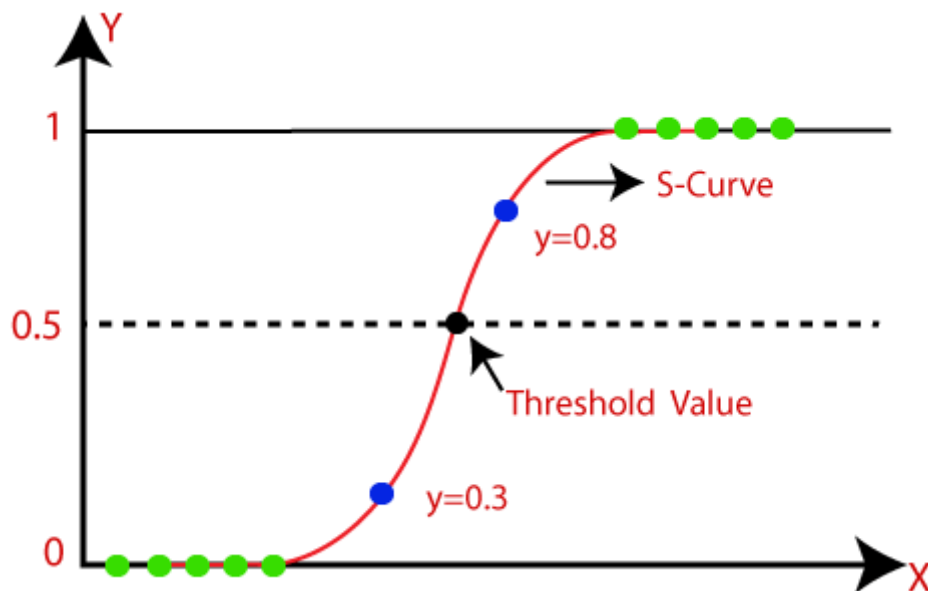


Fig 3.11

3.3.11 Naïve Bayes

In statistics, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features. They are among the simplest Bayesian network models, but coupled with kernel density estimation, they can achieve high accuracy levels.

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics literature, naive Bayes models are known under a variety of names, including simple Bayes and independence Bayes. All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a Bayesian method.

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

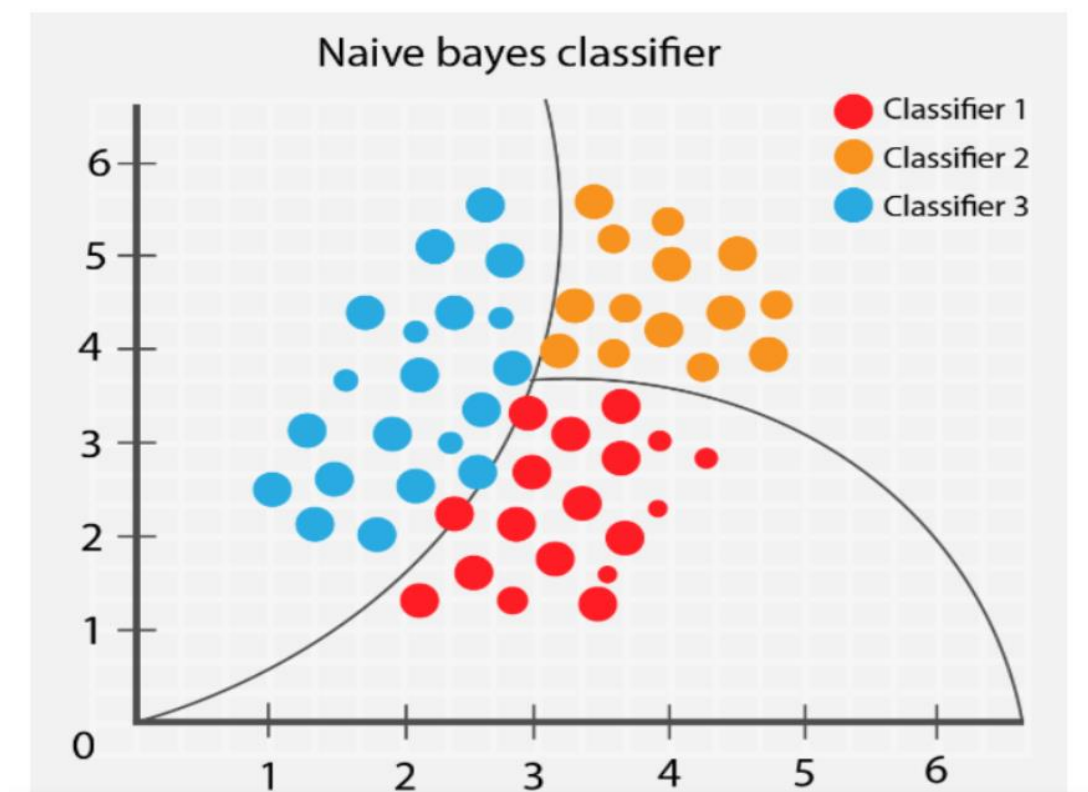


Fig 3.12

In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations.

In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers.

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

3.3.12 Convolutional Neural Network (CNN)

The structure of the visual cortex in mammalian brains was a pattern for creating the Convolutional Neural Network (CNN) [19]. The local pixel arrangement determines the recognition of the shape of the object, so CNN begins to analyse the image with smaller local patterns, and then combines them into more complex shapes. The effectiveness of CNN in recognizing objects in images can be the basis for solving the problem of speech recognition. There is a picture at CNN and its architecture reflects this. Thus, our method uses sound encoding using images. A typical CNN network includes convolution layers, pooling layers, and fully connected layers. Convolution layers and pooling layers constitute internal structure, and fully connected layers are responsible for generating class probability.

Convolutional Layer

Convolutional layer (CL) consists of neurons connected to the receptive field of the previous layer. The neurons in the same feature map share the same weights [20]. CL contains a set of learnable filters. One filter activates when a specific shape or blob of colour occurs within a local area [20]. Each CL has multiple filters, while the filter is a set of learnable weights corresponding to the neurons in the previous layer

Pooling Layer

Pooling layer (PL) is an effective way of nonlinear down-sampling. It has as the CL receptive field and stride; however, it is not adding any learnable parameters. PL is usually put after the CL. The receptive field of neuron in PL is two-dimensional. Max-PL is the most frequently used PL. Each neuron outputs the maximum of its receptive field. Usually, the stride is the same as the size of the receptive field. The receptive fields do not overlap but touch.

Back-Propagation

The single evaluation is completely consistent with the feed-forward neural network. The input data or activations are passed to the next layers, the dot

product is computed over which activation function is applied. Down-sampling the network using PL might be present. At the end, two or three fully connected layers are stacked. In order to use the gradient descent learning algorithm, the gradient must be computed. The usual back-propagation algorithm is applied with two technical updates. The classical back-propagation algorithm would calculate different partial derivatives of weights belonging to the neurons in the same filter; however, these must stay the same. Therefore, derivatives of loss function with respect to weights of neurons belonging to the same feature map are added up together. The update of back-propagation itself is when dealing with max-PLs. The back propagating error is routed only to those neurons that have not been filtered with max-pooling. It is usual to track indices of kept neurons during forward propagation to speed up the back-propagation.

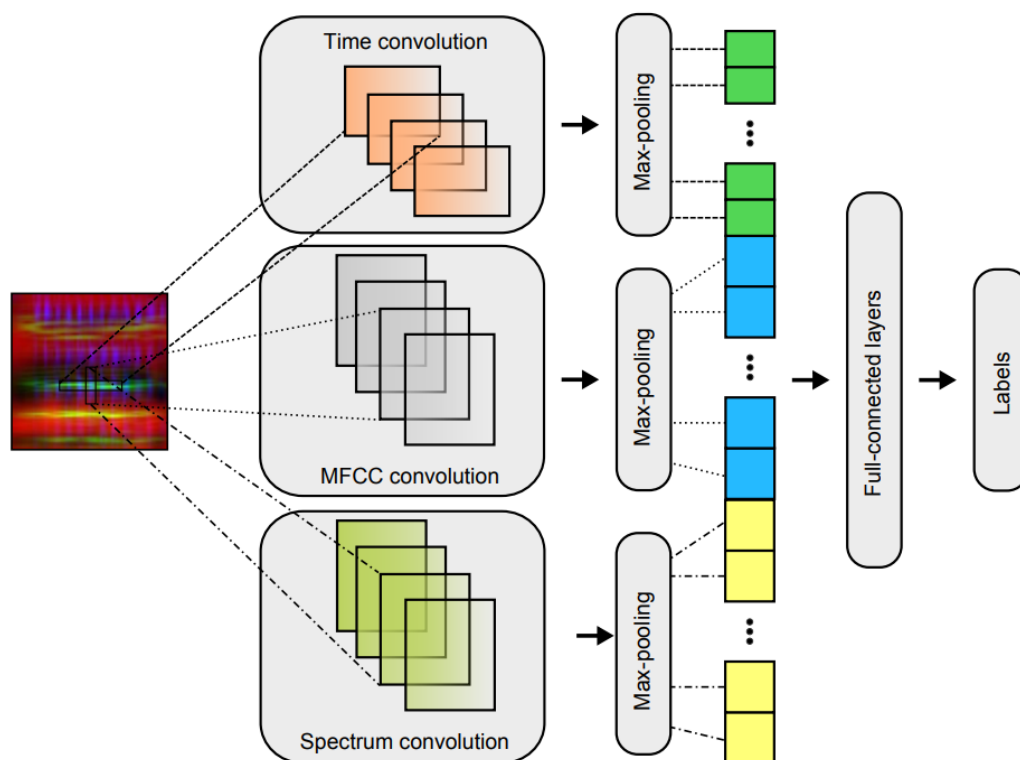


fig 3.13 Block diagram showing time-frequency convolution neural nets

The studies included two important issues. The first concerned the correctness of the work of the algorithm, realizing the division of speech into syllables. A method was applied that takes into account signal energy and signal frequency for the stationary fragment under consideration. The division into individual syllables was made for the experimentally selected threshold values. Each syllable pattern has been saved as an image in a directory name corresponding

to the designated syllable. The experiment was carried out for insulated words and continuous speech. A different number of words have been adopted for continuous speech. The most common in the literature are works that use time and frequency characteristics for analysis. Our work includes a new component in MFCC signal analysis.

3.4 Web Application Implementation of the models

After generation and training of the models, we created a web application for simpler access and use of the same. This web application was created using the Django python framework as it provided a consistent coding environment to the model files as both environments use python language at the base.

Django is a is a back-end server side web framework, that encourages rapid development and clean, pragmatic design, it is a free and open source framework. Django also supports a built-in sqlite database system which helps in storage application without the need for a new language or framework in the form of database applications.

Below is the structure of the final web application:

1. The Home Page:

This is the page which is seen by the users when they visit the website, this page allows the user to select the model which they want wish to use for their selected audio files, if no model is selected, the website automatically selects Cat Boost Classifier as the default model for the operation. Below is a screenshot depicting the structure of the page:

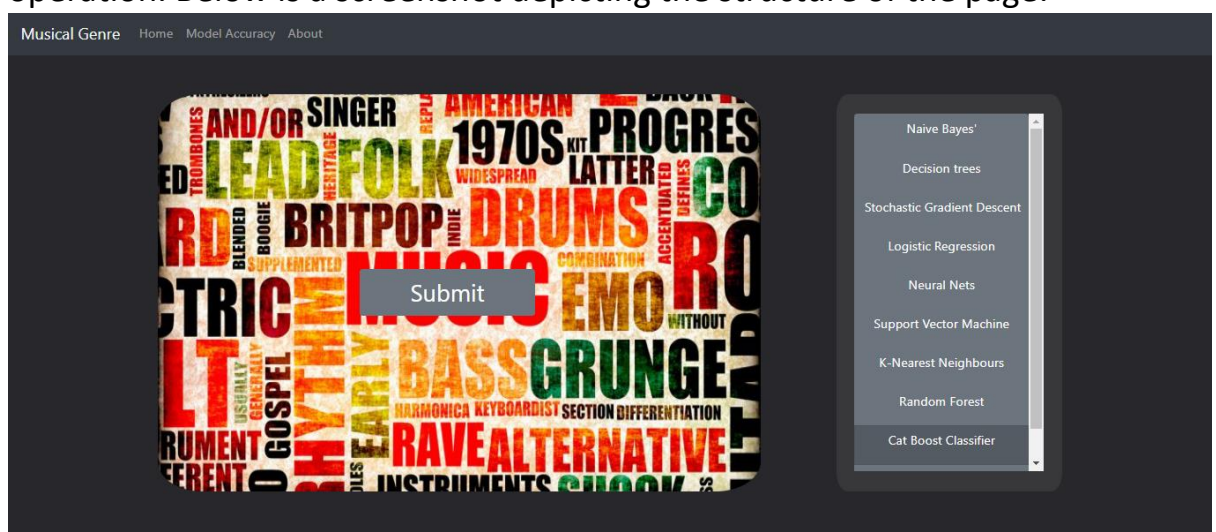


Fig 3.14 Home Page

2. The File Upload Page:

Once the user selects the model from the homepage and click on the Submit button, they are directed to the file upload page where they need to select the file they want to perform the detection operation on and give it an optional custom title, in case a custom title is not provided, the file name is asset as the title by default. Below is the structure for the file upload page:

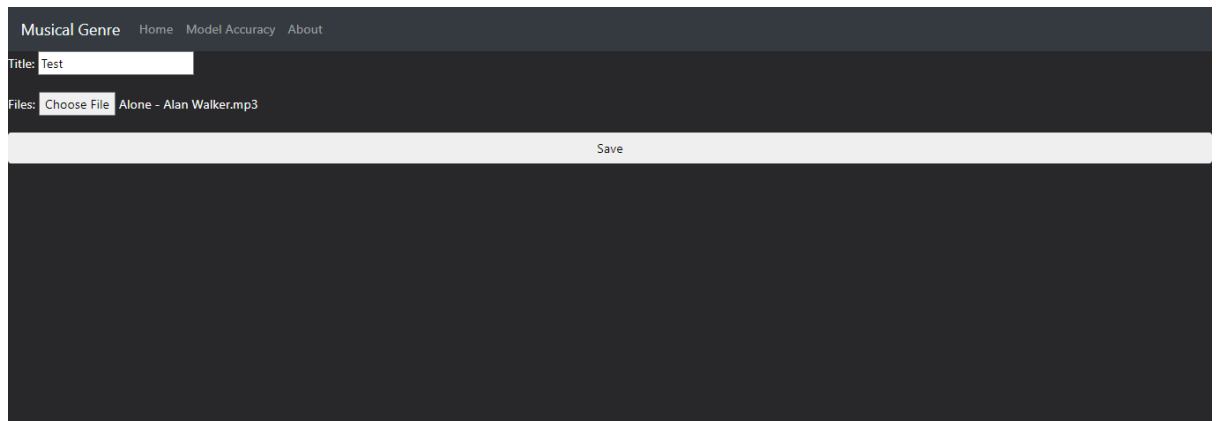
The screenshot shows a web application interface for file upload. At the top, there is a dark navigation bar with the text 'Musical Genre' and links for 'Home', 'Model Accuracy', and 'About'. Below this, there is a form area. On the left, there is a 'Title:' label followed by a text input field containing the word 'Test'. Below the title field, there is a 'Files:' label followed by a 'Choose File' button and the text 'Alone - Alan Walker.mp3'. To the right of the file selection area, there is a 'Save' button. The main body of the page is a large, empty dark rectangle.

Fig 3.15 File Upload Page

3. The Results Page:

After file is uploaded and the user clicks on the save button, the file is sent to the database along with the Title and the model selected by the user as parameters, based on the parameters the model file is selected and the detection algorithm is run on the file after pre-processing (trimming, feature extraction etc.) after the successful run of the model it also generates the Waveplot and Spectrogram of the audio file, then the user is directed to the Results Page where the Waveplot and Spectrogram of the audio file along with the final output from the model is displayed along with its accuracy. Below is the structure of the final results page:

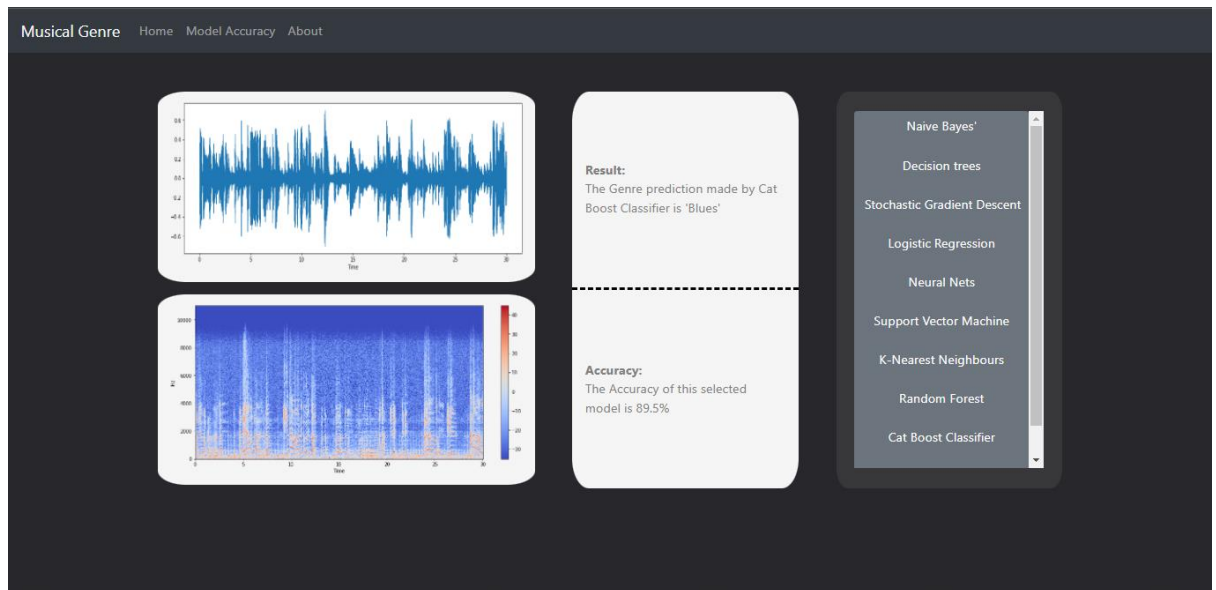


Fig 3.16 Results Page

This website is an extremely crucial part of our project as it helps us verify our models on a very large scale i.e., the entire public domain, it also helps our models to learn and improve their accuracy every time they're used for a new file and last but not the least this model helps us expand and enrich our data set with many more audio tracks which in turn gives our models more experience and increases their accuracy.

3.5 Outline of chapter 4:

In the next chapter we will see the result and accuracy of each and every model. In order to choose the optimum model for the classification of different genre of sample audio songs.

Chapter 4

Results and Analysis

This chapter displays the results in the form of highest accuracies achieved for each experiment, as well as confusion matrices to help visualise classification accuracies for individual genres. A discussion of the results also follows.

4.1 Discussion

4.1.1 Genre

The average genre scores across the test dataset of size 2997 were as follows: Blues: 188 , Classical: 270 , Country:198 , Disco:201 , Hiphop:207 , Jazz:229 , Metal:280 , Pop:263 , Reggae:212 , Rock:147.

Although performance varied with each classification algorithm, similar trends were identified in the results of each. For example, across most experiments, Metal , Classical and Pop were classified most accurately. Metal may have been one of the most successfully classified genres because its average scores for a number of the features were significantly different to those of other genres. This implies the presence of sonic differences that were detected successfully in the training of the algorithms. Hip Hop's place as one of the more accurately classified genre is harder to explain; it may be that the presence of rapped vocal passages (i.e. vocal passages that are not sung) helped differentiate it, if such passages led to distinctive feature trends. Further investigation of the dataset would be required to understand why Hip-Hop was classified the best.

Across all experiments, Rock was misclassified more than any other genre by a significant margin. Although accuracy was notably higher than random chance (10%), its final average accuracy was nearly half of that of the best-classified genre (Metal). It is unclear exactly why this is the case. A potential explanation may lie in the fact that as a concept, 'Rock' music may be more broadly and ambiguously defined than the other genres in the dataset. Perhaps the curators of the original dataset were working with a definition of 'Rock' that was looser than their definitions of the other genres. However, it is worth noting that Pop also encompasses a reasonably broad range of production styles, and yet it achieved one of the highest score of all the genres. Further investigation into the dataset is needed.

4.1.2 Output Format:

Presentation of output is very important in comparative studies. We output the results in following metrics:

Confusion Matrix:

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa.

4.1.3 Outputs of Models Trained

For all the confusion matrices, the rows are in the order Blues , Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae, Rock.

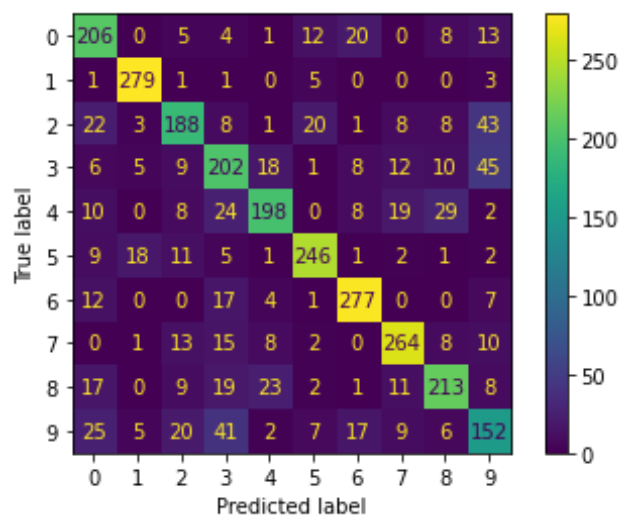


Fig 4.1 Confusion Matrix for SVM

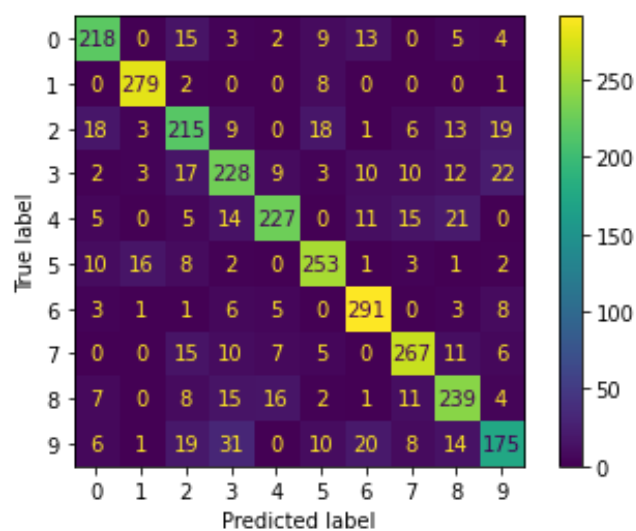


Fig 4.2 Confusion Matrix for Random Forest

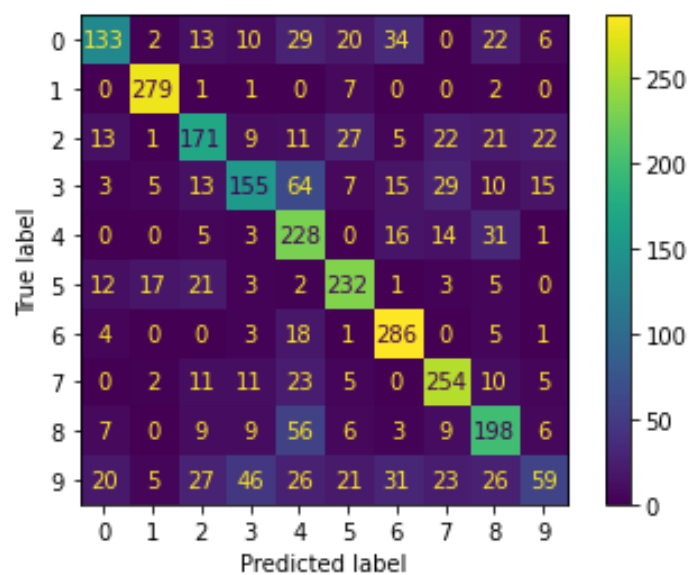


Fig 4.3 Confusion Matrix for Stochastic gradient descent

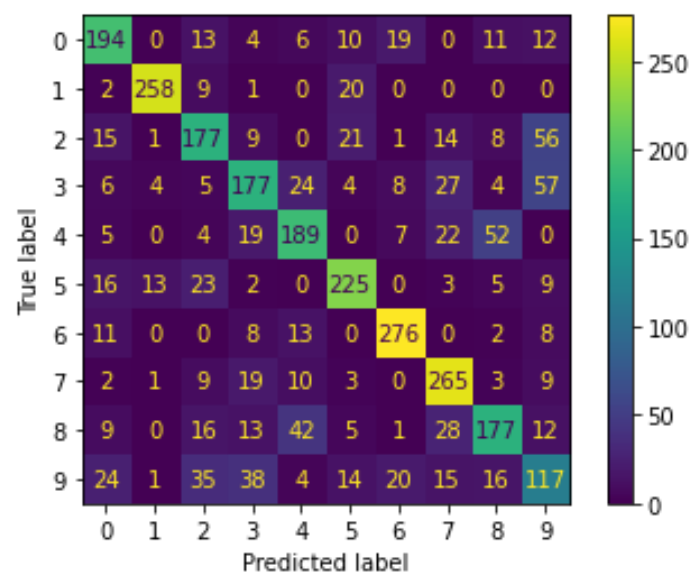


Fig 4.4 Confusion Matrix for Neural Networks

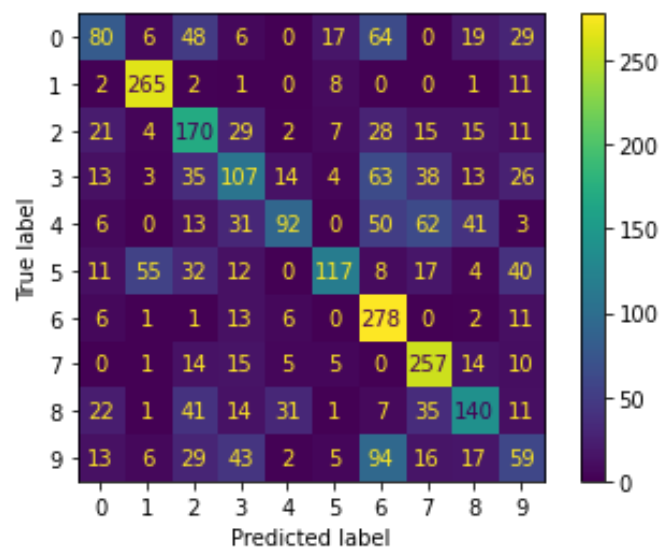


Fig 4.5 Confusion Matrix for Naïve Bayes

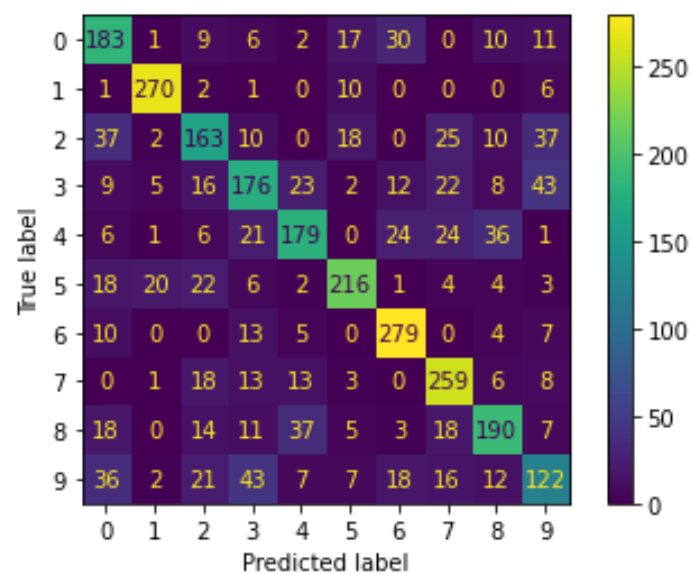


Fig 4.6 Confusion Matrix for Logistic Regression

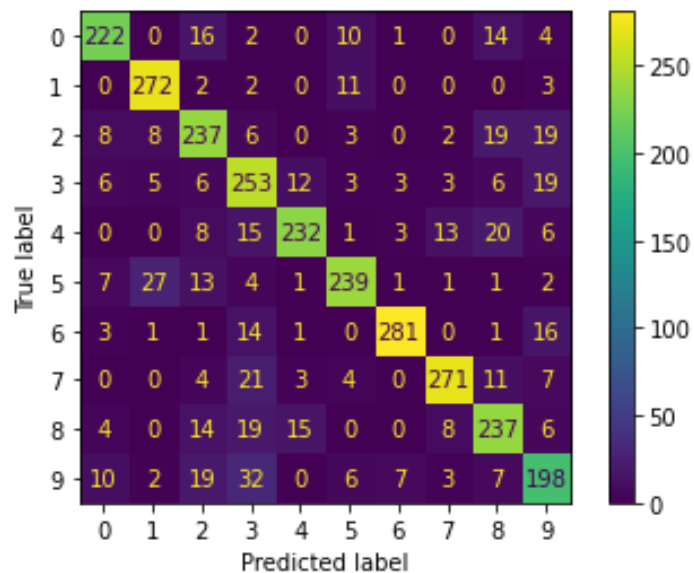


Fig 4.7 Confusion Matrix for KNN

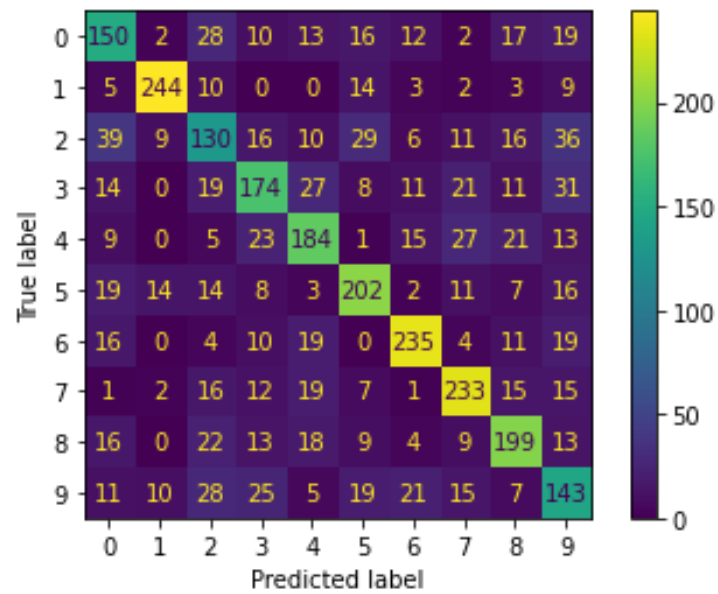


Fig 4.8 Confusion Matrix for Decision Trees

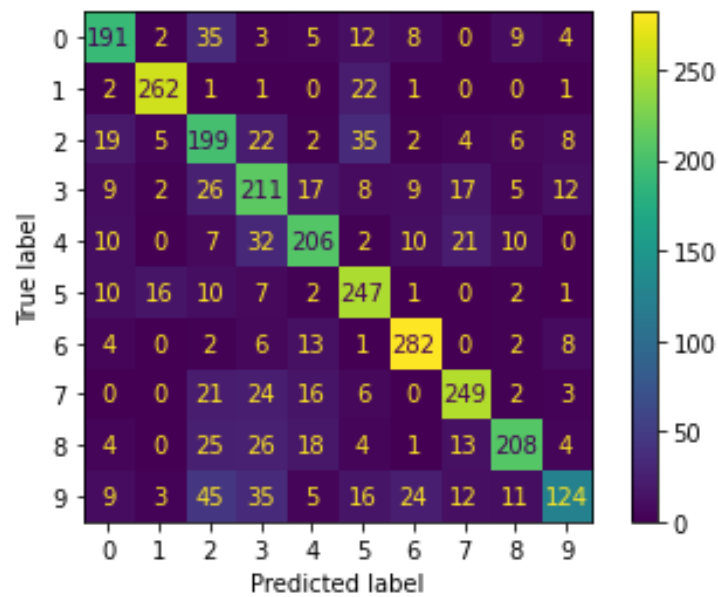


Fig 4.9 Confusion Matrix for Cross Gradient Booster (Random Forest)

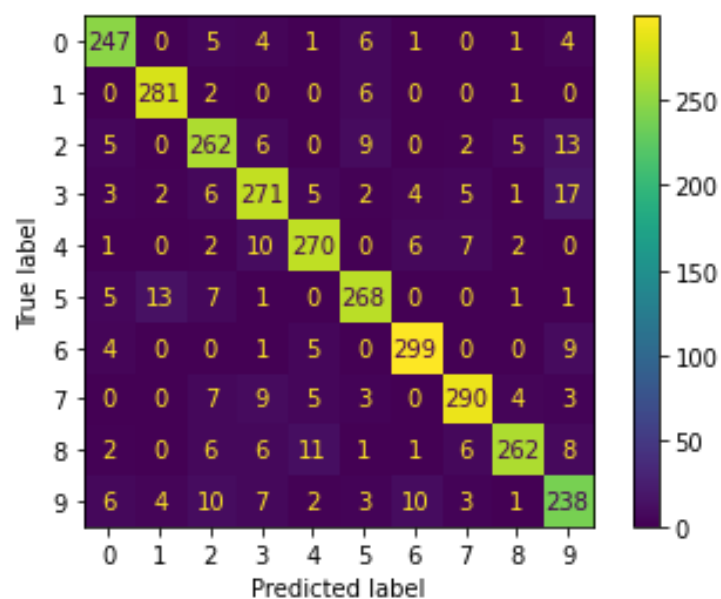


Fig 4.10 Confusion Matrix for Cross Gradient Boost

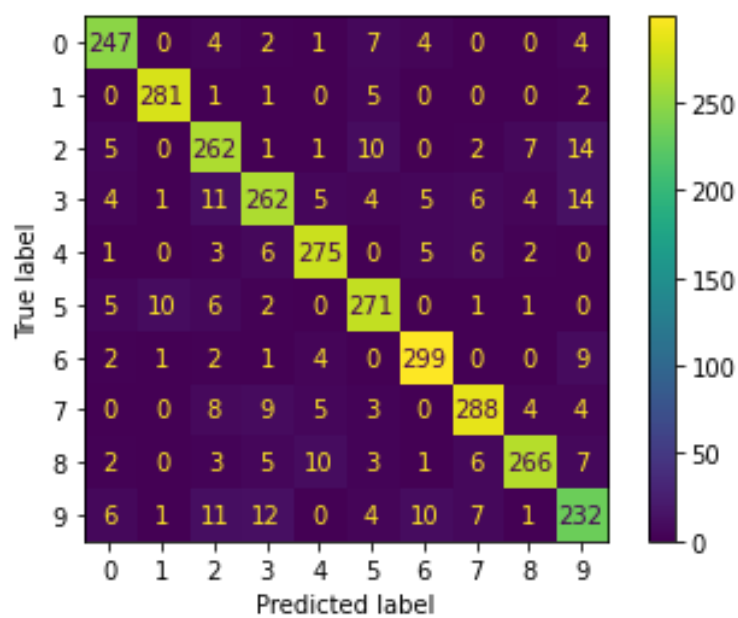


Fig 4.11 Confusion Matrix for Cat Boost

4.1.4 Comparison between Algorithms

This project initially set out to compare machine learning algorithms in terms of how suited they were to the problem of music genre classification. In terms of accuracy, on the standard dataset, the final ranking from best to worst is:

1. Cross Gradient Booster : 89.69%
2. CatBoost Classifier : 89.52%
3. CNN Model : 89.04%
4. K-Nearest Neighbour: 81.48%
5. Random Forest: 79.81%
6. Support Vector Machines: 74.24%
7. CrossGradientBoosterRandomForest: 72.70%
8. Neural Networks: 68.56%
9. Logistic Regression: 67.96%
10. Stochastic Gradient Descent: 66.56%
11. Decision Trees: 63.19%
12. Naive Bayes: 52.21%

The gradient boosting algorithm achieved the highest accuracy as can also be seen from the confusion matrix. The K-Nearest Neighbour Algorithm and Random Forest Algorithm also achieved a higher accuracy.

The gradient boosting machine outperformed the random forest, on both the standard dataset and the augmented dataset. This is in line with previous findings that suggest that gradient boosting machines may be more accurate than random forests under a range of conditions, although, of course, this will depend on the classification problem in question, and the random forest algorithm is still worth testing as it may outperform a gradient boosting machine in certain circumstances.

The neural network did not perform especially well in comparison to the other classifiers. The exact cause of this is unclear; it is true that, in general, neural networks offer more 'customizability' than the other classifiers discussed. This means that there are many more ways a neural network can be parameterized and structured by researchers; the cost of this may be that it is harder to structure one optimally, as there are more factors to take into account. Deep learning is typically performed on datasets with more features than were

present for this experiment, which may explain why it failed to outperform the other algorithms in this instance.

Overall, the results indicate that the Gradient Boosting Machine was the most promising algorithm of all the algorithms considered in this study.

Chapter 5

Conclusions

This study extensively compared machine learning algorithms in their suitedness to the task of music genre classification by using multiple datasets comprising of thousands of various data files of every genre for each algorithm, carefully mapping out their accuracy and observing them by using concepts like confusion matrices for every dataset and algorithm separately.

We used the confusion matrices of these models to find their prediction accuracy for each of the classes. We saw that rock had the least probability of being classified correctly by almost all models while metal songs were the most easily classified and most accurately as well, using the features extracted from the audio file.

We found initially that cat boosting algorithm that works on symmetric decision trees, cross gradient boosting algorithm that works in asymmetric decision trees, and convolutional neural networks provided the highest probability of classifying the audio file correctly.

Of the approaches surveyed, the gradient boosting machine learning algorithm was shown to be the most promising algorithm when computed independently by our classification program.

Classification of audio files was based upon 56 features extracted from each song of the used datasets, most of which pertained to spectral information

Future Scope

Several avenues for further investigation present themselves. First, as has been noted, the use of time-based features could allow for classification based on more complex musical properties, as could be performed by algorithms such as recurrent neural networks.

Second, other types of feature could be investigated in terms of their relevance for this type of classification problem. Only one explicitly rhythmic feature was extracted for this study - average tempo - thus greater emphasis on rhythm may represent an interesting opportunity for researchers in this field.

Key signature and melody could also be worthy of greater focus. Finally, more work on curating high-quality datasets could be performed. This could involve working with and reducing the size of pre-existing datasets, combining existing datasets together, or the collection of new tracks that are not currently in the public domain.

Overall, music genre classification remains an interesting and worthwhile challenge for both academic institutions and businesses alike, and there is plenty of room for further study and analysis.

References

- Y LeCun, L Bottou, Y Bengio, and P Haffner. Gradient-based learning applied to document recognition. In Proc. of the IEEE, 1998
- Tzanetakis, G. and Cook, P. (2002). "Musical genre classification of audio signals." IEEE Transactions on Speech and Audio Processing, 10(5), pp.293-302.
- McCallum, Andrew. "Graphical Models, Lecture2: Bayesian Network Representation". Retrieved 22 October 2019.
- Ng, Andrew Y.; Jordan, Michael I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. NIPS. Vol. 14.
- Brownlee, J. (2019a). A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning, Machine Learning Mastery. [online].
- Brownlee, J. (2019b). Metrics To Evaluate Machine Learning Algorithms in Python, Machine Learning Mastery. [online].
- Chang, C., Lin, C. (2013). LIBSVM: A Library for Support Vector Machines, ACM Transactions on Intelligent Systems and Technology 2.
- Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique, Journal of Artificial Intelligence Research 16 (2002), AI Access Foundation , USA, pp. 321–357.
- Defferrard, M., Benzi, K., Vandergheynst, P. and Bresson, X. (2017a). 'FMA: A Dataset For Music Analysis', ISMIR 2017 - 18th International Society for Music Information Retrieval Conference, Suzhou, China, 23-27 October.
- Jensen, J. H., Christensen, M. G., Murthi, M. H. and Jensen, S. H. (2006). 'Evaluation of MFCC estimation techniques for music similarity', 14th European Signal Processing Conference, Florence, Italy, 4-8 September.
- Kopczyk, D. (2019). From Decision Trees to Gradient Boosting. [online]. (Published 18 April 2019).
- Li, H., Xue, S. and Zhang, J. (2018). Combining CNN and Classical Algorithms for Music Genre Classification. Institute for Computational Mathematical Engineering, Stanford University.
- N Scaringella, G Zoia, and D Mlynek. Automatic genre classification of music content: a survey. IEEE Signal Processing Magazine, 2006

Kubanek M, Bobulski J, Kulawik J. A Method of Speech Coding for Speech Recognition Using a Convolutional Neural Network. *Symmetry*.

G. Riccardi and D. Hakkani-Tur, "Active learning: theory and applications to automatic speech recognition," in IEEE Transactions on Speech and Audio Processing, vol. 13, no. 4, pp. 504-511, July 2005, doi: 10.1109/TSA.2005.848882.