Cardiff School of Computer Science & Informatics

One semester individual project

Final report

40 Credits

Sample based guitar transcription

Author:

Alec Manders

Supervisor:

Professor. A D Marshall

May 5th

2015

Acknowledgements:

Thanks to A. D. Marshall for his continued support and advice throughout such an interesting project.

Contents

1. Introduction:

Guitar music has dominated popular culture for the past 50 years, because of this there are millions of guitar players internationally; however the technologies utilized by guitar players to help learn and understand music has progressed little since the 1950s. Tablature has become more available through mediums such as the internet yet for most people playing the guitar still involves searching for a source of transcribed guitar music that they can then learn from. This project proposes a technology that can not only change the way that people learn other people's music, but aid them in the composition and recollection of their own pieces of music as well as tell what kind of guitar is being played on a particular piece of music.

The purpose of this project was to produce a system that would be able to differentiate between features of guitar audio, and then be able to use those discriminations to provide greater insight into the characteristics of the piece of recorded guitar audio through a machine learning system. Knowledge of how guitar audio works has been developed throughout this project through a variety of research and testing methods, this knowledge has then been utilized to extract relevant information from guitar audio samples. In a previous related project it had been proven that this was possible, the aim was that my project would be a continuation of these ideas that would result in the description of a much more detailed system, and aim to prove how effective it could be at certain types of transcription.

Before research for this project started it was assumed that although the same notes can be played in different places on the guitar's neck, by analysing these notes it would be that there would be a discernable difference between two notes being played in those different positions. Machine learning seemed the most appropriate method to take to achieve this goal due to the nature of having to compare one piece of data to large numbers of other pieces of data to see how it should be classified. For this project thorough examination of the spectral domain was used to gain the appropriate data needed to detail whether or not an effective system could be made.

2. Project Steps:

As with any project this one could be broken down into a number of steps that had to be completed before progression could occur. Due to this project assimilating many ideas from previous projects and papers there was a certain amount of understanding to be gained before the project could progress.

- Initial planning was required. A time scale was detailed so that certain targets would be met. This was considered essential as the time scale was limited for a project with such a large scope.
- Research into the methods that had been previously used to achieve related goals in past projects was undertaken, these included in-depth research into machine learning and the characteristics of guitar audio.
- Code was obtained to extract information from guitar audio that could be used to classify its characteristics, this code then had to be studied to find out how it was achieving its results.
- Code was developed to convert data into a suitable format so that it could be classified correctly.
- Code was developed so that data could be farmed more easily from certain sources.
- Large amounts of relevant data was farmed from a variety of sources, this data was then manipulated and sorted so it could be more easily classified.
- A range of classifiers were developed to differentiate between different guitar types, the different positions in which a note could be played and expressive techniques being played on the notes, for example vibrato or string bends.

2.1 Outcomes from this project:

- The proof or lack of proof for a previously outlined method for finding the string and fret that an audio sample is being played on.
- A method of farming audio data from a MIDI input source.
- A selection of classifiers that can differentiate between a range of different guitars and could be used to aid in the determination of where on the guitars neck a note is being played.
- A tool that can be used to remove single audio files containing one note from a long audio file, which can then be passed into a system to be transcribed.

3. Background:

3.1 Initial Research:

3.1.1 Machine Learning:

After the realisation that machine learning was to be used to classify the different aspects of a piece of guitar audio (type of guitar it is being played on, note and fret position, etc.) some detailed research into the subject had to be undertaken. Having had almost no knowledge of machine learning prior to this project I became dedicated to understanding the fundamental principles that I could utilize to achieve my goals. After a little independent research online I discovered that it was possible to take a free online course at Stanford University on machine learning taught by Andrew Ng (Ng, 2014). This course was both extremely informative and accessible and I would highly recommend it to anyone interested in the subject of machine learning. The course covered both supervised and unsupervised learning as well as detail about cost functions. Although much of this information was not utilized in this project I feel that progression would have been much slower without the use of this online course, information about clustering and linear regression was extremely helpful. Another main bonus of taking this online course was that it used the LibSVM toolbox (Chih-Chung Chang, 2014) which was suggested to me by my project supervisor. Being able to watch someone else use the LibSVM toolbox made it much easier for me to understand the process and assured me that it was the correct tool for the job. By undergoing this course I felt that I was greater prepared for any issues that I might encounter as it detailed common issues that occur during classification such as over fitting (when the machine learning algorithm catches the noise of the data) and under fitting (when the machine learning algorithm cannot find a trend in the data).

3.1.2 Extracting Feature vectors:

The process of utilizing relevant data from guitar samples to create feature vectors was to utilize a method which was defined in a paper titled "Signal representation and estimation of spectral parameter by inharmonic comb filters with application to the piano" (Alexander Galembo, 1999). The purpose of this paper is to detail a method in which the fundamental frequency of a recorded piano note can be measured and the inharmonicity coefficient can be found or to present "A convenient representation of how much a spectrum deviates from the strictly harmonic case ("degree of inharmonicity")". The harmonic case refers to the harmonic series where the wavelength of the overtones of a vibrating string are 1/2, 1/3, 1/4 etc., of the strings fundamental wavelength. The method of extracting feature vectors is based on using inharmonic comb filters which filters out bands non-equidistantly, based on a function and that calculates next step in the harmonic series while assuming progression through the harmonic series will not be exact due to variations in string type; the result can then be extracted as feature vectors that can be used to identify the audio. The inharmonicity coefficient would be based on the dimensions of the string upon which the note was being played and the young's modulus of the string. The idea being that by analysing the partials defined by the filter bands of the inharmonic comb filters you would be able to get a reasonable estimation of the fundamental frequency of the note as well as a feature vector unique to that note position.

3.2 Existing Products/ Marketability:

There are currently several products that have the ability to detect the frequency of notes being played and even in certain situations chords that are being played, mostly this software exists within electronic guitar tuning software. Despite these products there doesn't seem to be any software that will do a satisfactory job of transcribing guitar, or any kind of music, for the user automatically. As well as this I have been unable to find a product that has the ability to differentiate between different types of guitar and plucked intensity. One of the more popular pieces of software used to transcribe is known as Capo 3 (Capo, 2015) although I have not used this software personally the reviews are not entirely positive and it does not offer any form single note prediction. Despite this Capo 3 does offer a wide range of features outside chord detection.

The popularity of electronic music production continues to sky rocket, an example of this being that the only musical genre on USA to achieve growth in track sales is dance/electronic music, which had an increase of 8% over one year where as sales of rock track sales declined by 12% (Watson, 2014), although this could potentially be defined as a fad, the evidence is still surprising. Because I believe a piece of software that can correctly solve the problems which I have attempted to solve could be commercially viable, and having spent many hours producing music via the computer myself, I can say that being able to transcribe something that I have played on the guitar quickly, that I could then put easily into a music production suite such as Ableton (Ableton, 2015) would be very appealing. An Exact figure for the number of guitar players in the world has been hard to obtain, however I feel it is safe to assume that due to the dominance of the guitar in popular music in the past 60 years, it can safely be assumed that guitars are one of the most commonly played instruments internationally.

4. Applications for this technology:

There are a select number of groups that would be interested in this type of product and the research that this project utilizes, however these groups potentially contain large numbers of people. Currently the features that have been developed in this project and previous projects would only really be useful for guitar players who may wish to record themselves whilst playing and have their music transcribed for later use. Due to the nature of how this system works there is a requirement that there is very limited background noise present therefore trying to transcribe guitar music with instruments playing in the background would be difficult. I also believe it is entirely possible that this system could be applied to any stringed instruments that have the same characteristics that define harmonicity as a guitar does. Transcribing music is currently a long and arduous process and amateur guitar players rarely transcribe their own music and can often forget interesting refrains. Although my research does not currently include certain expressive techniques such as bends, hammer-ons and slides, the research conducted in this project suggests that with enough time it would be possible for these to be recognised through the collection of more high quality annotated data, the lack of which has been a recurring issue though out this project

Guitar transcription of successful pop songs is very common on forum sites such as ultimateguitar.com, however one issue that many people who transcribe music have is deciding where on the neck the riff or chord is being played. Although the work that has been done will not be applicable to music containing anything other than one instrument certain parts of songs such as solos, and genres of music that are purely guitar based would become much easier to transcribe.

4.1 Music Lessons:

The teaching of guitar is a common profession for many musicians globally, the successful application of this kind of software to guitar lessons will help engage and enrapture students due to the ability to see pieces of music they have written come to life in the form of tablature. My belief is that musicians will find it immensely rewarding to have a piece of music that they have developed committed to paper so that they and others can continue to play it in the future.

4.2 Gaming:

Games that include a simulated guitar playing experience such as Guitar Hero and Rock Band have proved best sellers over the last 15 years, with many featuring a variety of instruments. The Guitar Hero franchise alone claims to have sold over 25 million units, with Guitar Hero 3: Legends of Rock grossing 831,000,000 since its release (Reisinger, 2011). These games work on the premise of pressing one of 5 coloured buttons in time with the screen and strumming the paddle at the correct moment. I feel that with sufficient development it would be possible to create a game that could be used to aid the teaching of guitar that works on the same principle as playing along with the screen. Many people devote significant time to playing these simplified guitar emulators, if a similar amount of time was spent playing similar games with real guitars we would have a large global community of virtuoso guitar players.

5. Approach:

5.1 Tools:

Defining the correct toolset at the beginning of this project was very important. In terms of programming language it was essential that a language was chosen that could perform all of the required tasks. The option for using multiple programming languages was considered however it was concluded that this would only increase the difficulty of the implementation.

5.1.1 MATLAB:

MATLAB was selected as the programming language for this project. All code was designed in MATLAB and all machine learning and MIDI manipulation tasks were achieved through MATLAB. There were several reasons for choosing MATLAB which included:

- MATLAB includes a wide range of digital signal processing (DSP) tools, it is very easy to read in and manipulate wav files as well as visualize waveforms, and perform transforms into the spectral domain such as the Fourier transform. Being able to visualise the sample rate and sampled data in an array in MATLAB made implementing certain functions immeasurably easier.

- MATLAB, like many other programming languages includes a way to easily import open source packages, there were two packages available for MATLAB without which this project would not have been possible, Ken Shutte's MIDI scripts (Schutte, 2012), and LibSVM (Chih-Chung Chang, 2014).

- A project of this scale could have been programmed from the start in languages such as Java or C++ however the use of MATLAB's existing packages and features allowed for the rapid development of this project. If this research gained from this project were ever to be commercialised, then it will almost certainly have to be reprogrammed in Java or C++.

- Although this project provides some knowledge that can be used to aid later projects it was fundamentally an educational exercise. I decided that MATLAB would be a good choice of language as although it was not completely alien to me, I was by no means a proficient MATLAB programmer. I reasoned that gaining more experience in MATLAB programming would be valuable to me in the future, having previously only had any real knowledge of Java and Python and I am aware that experienced MATLAB programmers are highly sort after.

5.1.2 LibSVM:

It rapidly became apparent that this project was going to need to involve supervised machine learning, a technique whereby you feed a machine a set of data with known characteristics, and the machine then attempts to classify unlabelled data based on the information the machine has previously learned. It was assumed that guitar data would be widely available in both commercial and home recording packages. It was also assumed that this data would be named according to a suitable set of rules and therefore supervised learning was used for classification. Support vector machines were also a standout choice for

this project as they facilitate multi-class classification which would prove vital. LibSVM (Chih-Chung Chang, 2014) was chosen as the classifier for this project upon following the recommendation of my project supervisor. LibSVM contains a number of built in functions that allow for the easy creation of classifiers and the writing of data into a format in which it can easily be classified. There are multiple clustering methods that can be used for LibSVM including linear regression and radial, its versatility being one of the reasons it was chosen for this project. LibSVM also offers several ways to view information about clustered data such as prediction accuracy as well as providing effective methods of cross validation.

LibSVM is also supported in a large range of other programming languages, including many of the more popular languages such as C++, Java and Python. If reprogramming in a different programming language were to occur, using LibSVM would improve the portability of the work I have done.

5.1.3 Ken Schutte MIDI scripts:

The original plan was to use a widely known MIDI Toolbox (Toiviainen, 2004) to read in manipulate MIDI files, however having examined the MIDI toolbox features it was decided that many of them would overcomplicate the problem. As an alternative it was decided that a set of MATLAB functions to read in midi files created by Ken Schutte (Schutte, 2012) would be used. These MATLAB scripts proved simple to use due to the detailed documentation on the website and they provided all the functionality that was needed to properly edit and control the MIDI files. The "midiInfo" script that returned information about the 8 MIDI channels was a very effective way of visualising how information is stored in a MIDI format and made it simple to manipulate the MIDI files corresponding audio files.

5.2 Problems:

One of the main problems needing to be overcome was the collection of properly annotated guitar samples. For much of the project's timespan this was an issue, however my project supervisor utilized a Boss GP-10 guitar processer (Boss, 2015) to generate a selection of wav files that could be manipulated via their corresponding MIDI files. The Boss GP-10 is a guitar processor that amongst other things allows for the emulation of a variety of different models of guitar. This made it possible to create classifiers for many more guitar types in addition to creating distinct note classification tests for different types of guitar. Some of the features of the Boss GP-10 include being able to play in a wide range of tunings, emulation a set of guitar models and different pickup types. It is also possible to create your own models and apply a huge range of effects to the audio. One issue that may occur with this is the accuracy of the representation of the different guitar models compared to real recording of those models. The human ear can hear that there is a similarity to the type of guitar that is being emulated however due to a lack of proper data it has not been possible to test real recordings of these guitar models against the emulated versions to see if accurate classification occurs.

6. Features:

The purpose of this project was to prove if a certain method was possible whilst simultaneously creating a set of features that could be used to build a fully functioning transcription system in the future. Below is an explanation of all the features that have been implemented in this project and what part each feature would play in a fully functioning system.

6.1 Validating Models:

This project would have failed if it has not been possible to validate the accuracy of models created through LibSVM; without sufficient accuracy validation it is impossible to know whether or not your model has been successful.

For this project cross validation methods have been used. Cross validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set, this is achieved by using a portion of the data set that could have been used to train the model to be tested on the model after it has been created. For this project it was decided that repeated random subsampling would be used as a cross validation technique. The theory behind repeated random subsampling is that the training data is randomly split into training and testing data. For each split the predictive accuracy is assessed using the testing data. In this case 20% of the training data was used to create the testing set, as is often the case for this model.

This method of classification works very well with the type of data sets used during this project. After correctly labelled data was sufficiently organised it became very easy to then siphon off a portion of this data to use for testing data. By using these methods of model validation it was possible to create a wide range of classification models.

6.2 Guitar type classification:

The use of the Boss-GP 10 made it possible to create a range of classifiers for many different types of guitars and guitar like instruments, this list includes Stratocasters, Telecasters, Les Pauls, Martins, Banjos, 12 String electric and Sitars. It has been previously noted that matching against real recordings of these guitars has not been possible due to the lack of appropriate data.

It has also been possible to create distinct classification for electric, acoustic and "other" guitar types where other represents a fretted string instrument such as a Banjo or Sitar. Due to the machine learning nature of this project the ability to classify characteristics based on certain features of guitar audio is only limited by the amount of data available to the classifier.

6.3 Expressive Techniques:

It has been possible to build some version of a classifier for certain expressive techniques such as vibrato. The theory behind this is the same as all the other classifiers: if the difference can be noticed by the human ear in theory it can it be noticed by the classifier. The lack of data was again going to be a problem during this creation of this classifier, however this test was conducted with the knowledge that even if completely accurate results were not possible, it would at least provide insight into whether or not further research into this classifier would be worth attempting.

Vibrato tests were also conducted between notes on specific guitar types and vibrato notes played on a range of guitars; these were compared to simple notes played on a range of guitars without vibrato. All of the data used for vibrato testing was gained from the Boss-GP10 farmed data. A plucked intensity classifier was created during a previous project called "machine translation of guitar audio" (Cohen, 2014) however as the data farmed from the Boss-GP10 didn't not contain a clear set of plucked intensities this type of classification was not tested. Following further analysis of the results produced in that project and the results produced during this project it concluded that this type of classifier could be easily created with enough accurate data.

6.4 Exact Note Prediction:

The ultimate goal of this project was to prove the idea that machine learning and classifiers could correctly predict the position of a note being played on the guitar's neck, so that it could eventually lead to accurate transcription methods. This assumed that there would be a discernible difference in the harmonics of a feature vector for different notes in different positions, the main difference being that it would decease as the player moves up the fret board. There is a way of detecting pitch using the inharmonic comb filter (Pritchard, 2002) however this does not define where on the neck the note is being played, as on the guitar it is possible to play the same note on different strings, sometimes on up to 5 different strings. This means that using the inharmonic comb filter alone is not an effective method for guitar audio transcription. The inharmonic comb filter was originally developed on the theory proposed in "Signal representation and estimation of spectral parameter by inharmonic comb filters with application to the piano" (Alexander Galembo, 1999) , and for the purposes of piano transcription it would be effective considering each note can only be played in one position on a piano.

Previous projects have undertaken this task in the past and produced interesting results including some form of note prediction (Cohen, 2014), a problem encountered in previous projects is that many data sets of guitar audio do not contain the same note being played in many different position on the guitars neck, only one note in a position, that is more often than not unidentified. Due to this problem previous methods of note prediction have only been tested on a very small data set.

Due to the inclusion of datasets generated through use of the Boss-GP10 has been possible to create wav files of all the different positions on the neck that a note can be played, for a number of guitars. For greater prediction accuracy it would require the generation of larger data sets as well as the collection of a wider range of plucked intensity data. As this information has been recorded as MIDI it may be possible for later projects to simply edit the velocity values of the MIDI information and rewrite the data, this would save time in actually recording information and would make the plucked intensity data more coherent. It may also be possible to analyse the behaviour of the MIDI file when vibrato is applied to a note and create a method of adding vibrato to guitar recorded though MIDI without having to apply it during the initial recording.

6.5 Labelling of content and organisation of files:

Throughout this report a method is outlined by which it is possible to divide up MIDI files and create distinct wav files that can be used for classification in later similar projects. The method for doing this includes a way of naming which string the note is being played on, which guitar the note is being played on, what the exact note is and what octave it is in, for example "LesPaul_AString_C3". This also facilitates all tunings that are commonly used on standard electric and acoustic guitars. I feel this method of MIDI classification is very

important as technologies included in effects units such as the Boss GP-10 used during this project, are becoming much more advanced, and the price of such pieces of equipment is falling. By looking at trends in the market of music production it is expected that there will be a rise in people using such devices to record themselves directly to MIDI, with a direct correlation to the number of people using computers to generate music. Although software to automatically transcribe recorded guitar midi exists, including it within a package that also transcribes recordings without midi information would make it very appealing to users.

Code has also been developed that will enable the user to easily extract certain files from a long list of similar files that can be used for later classification. This works on the premise that most single guitar samples which can be downloaded will have some sort of naming system determining their characteristics. An example of this being the "Indiginus Acousitc Guitar Collection" (Indiginus, 2015) where files are named "TPA3SL" where TP relates to the type of guitar the note is being played on, A3 is the note being played and SL denotes that the note is being slid to.

6.6 Wav File Dissection:

To create a fully functioning system it will be necessary to have a way of dividing up a wav file to create singular audio files, each containing singular notes that can be passed through the system and then transcribed. During this project I have created a way of chopping the audio files into overlapping windows of a defined size, with an overlap that is a portion of the window size. For example for the current audio file I have been testing the script with the window size is 2 seconds and each window has a 1.5 second overlap as the window moves though the audio file 0.5 seconds at a time. These audio files are then processed using the inharmonic comb filter (Pritchard, 2002) pitch detection method and an average of a certain number of concurrent processed files is taken to gain an average prediction of the note. It is important to take an average prediction of a set of concurrent notes as due to the overlap many of the notes are put through the inharmonic comb filter more than once. The script that I have devised works in a sequential manner:

- The audio file is read into MATLAB and the files length is calculated using the length of the sample data array divided by the sample rate.
- A start time of the portion wav file to be tested for the pitch is defined, originally at the first element of the array, and the end time of the portion of the file you wish to test the pitch of is defined as the previously determined start time plus the length of the windows to be passed into the inharmonic comb filter.
- A portion of the wav file starting and ending at the points previously defined are written to a new wav file in a new directory and named accordingly.
- Each of the wav files in this directory is then run through the inharmonic comb filter and their predicted frequencies are saved into a new array.
- The pitch of these elements is then averaged every n elements, depending on the window and overlap size.
- A Hashmap containing the frequencies in Hz of all notes that can occur on a guitar and their corresponding note names is created.
- Each of the elements of the averaged frequency array is then compared to this Hashmap using an adaptation of the binary search algorithm and a closest corresponding frequency is displayed on MATLAB.

This wav slicing tool is not as yet fully optimised yet and a more effective way of detecting where the notes occur, such as various beat detection techniques using the

Fourier transform could be utilized. During testing although the results were not perfect, due to too many notes being outputted because of the number of over lapping windows, pitch prediction of all the notes in the audio file was very effective.

6.8 Functions

During this project as many of the MATLAB scripts as possible were made into functions. The reason for doing this is that it will allow future projects to pick up exactly where this project ended without having to rewrite any functions. In all of the functions a description of what the function does and an example of input values to be used is present.

Some of the scripts were unable to be turned into functions due to the amount of information that had to be changed each time they were ran, for example the classifier script that also creates ROC curves and confusion matrices however within these scripts comments about which values should be changed and what each line of code does exist.

7. Method/Implementation:

Although the ultimate goal of this project was a tool that could be used for effective guitar transcription it was essentially a research project to determine if a certain method could be used to achieve effective note prediction. Due to the nature of the method proposed, high quality guitar sample data is a huge necessity therefore the processing of data and the creation of new data that could be used for the creation of more effective classifiers had to take precedent over a system that actually produced an output of recorded guitar transcription. Due to the time constraints, the overall goal of this project became the creation of a set of tools that someone could utilize, and potentially design a user interface around. The aim was that by the end of the project methods that could be used to achieve a successful transcription method would have been investigated, and if proved successful, future projects would be able to effectively implement the ideas and if not, the idea could be abandoned.

7.1 Guitar Audio Data:

It has previously been noted that the Boss-GP10 guitar processor was used for the harvesting of guitar samples from a range of different guitar types including the same guitar types with vibrato applied to them. During this project a number of samples from other sources were also included. The project supervisor for this project granted access to his extensive guitar sample library, however many of these samples were designed to be used with plugins and without either buying the plugin or spending money on a product to convert these samples, many of them could not be used. Although not all the data was valid there was still enough sample data to create an effective classifier for electric and acoustic guitars.

For this project Three acoustic guitars and two electric guitars available to me were recorded, however due to a lack of proper recording equipment it was not possible to obtain a set of guitar samples without a level of background noise. It seems likely that in many real world situations users would encounter a similar problem, being able to develop a system that can work around this would be hugely beneficial. The a major disadvantage of recording this audio data myself was the amount of time required, recording each note on each fret of 4 guitars, naming and processing all the files was time consuming and this was not a particularly effective method of harvesting data.

The specific guitar data that was recorded via the Boss-GP emulations provided by Professor A D Marshall, supervisor for this project, included for 12 String, Telecaster, Stratocaster, Martin, Sitar, Banjo and Les Paul emulations:

- Full picked version of each fret and string on the guitar where the note rings out until it fades naturally.
- A damp picked version of each fret and string where the note was stuck and then quickly palm muted to achieve a shorter version of the note.
- Vibrato played on each fret and string on the guitar.
- A series of articulations that could be used for testing both classifications and how well the MIDI slicing tool works.

7.2 MIDI Data Farming:

The method of farming data from the Boss-GP10 meant that it would be possible to create a system that would allow the farming of data from any MIDI source in the future, currently the Boss-GP10 allows for the emulation of a wide range of guitar sounds that is constantly growing, it also allows for the emulation of specific pick-ups so the potential for creating a wide range of classifiers was vast. The main principle behind the farming of this data would utilize a selection of scripts created by Ken Schutte to gain information about the MIDI files. The actual MIDI information was gained by using a function called "midiInfo" which created an 8 by n array where channels:

1 = Track Number
2 = Channel Number
3 = Note Number (midi encoding of pitch)
4 = velocity
5 = Start time (seconds)
6 = end time (seconds)
7 = message number note on
8 = message number note off

By using this information it was possible to define where a note started and stopped and the pitch the note being played. Due to the nature of the recorded samples it was possible to see where string changes occurred due to large drops in the pitch of notes, this information also related to which channel the MIDI data had been recorded on, as there were 6 available channels it was possible to accurately name which string the note was being played on. By utilizing this information it became possible to chop each sample individually and by using a hash map to relate MIDI note numbers to chromatic scale values it became possible to name all of these files correctly as they were created, including a suitable prefix and suffix to help describe which instrument the BossGP10 was emulating and on which string.

Relating this information to the wav file required that the length of the wav file was computed based on its sample rate and number of bits (length of the bits array divided by the sample rate), information that is created when using the "wavread" tool in MATLAB. Once the information for the MIDI files and the wav files became comparable it became possible to calculate the exact times where the wav file should be split. Due to the need to divide the length of the wav file it was necessary to round the result so that it was pointing to an existing index in the array, rather than a fraction of a number. It was decided that rounding up or down to decide which element to split the audio file on made little difference in an audio sample data array of approximate length of 20 million elements. The code that was used to calculate the information needed to correctly split the wav file and the code that was used to the naming conventions of the strings is presented below (Figure 1, Figure 2).

```
for k = 1:lengthNotes
    i = [];
    startTime = notes(k,5);
    endTime = notes(k,6);
    splitStart = startTime * Fs;
    splitEnd = endTime * Fs;
    splitStartRound = round(splitStart);
    splitEndRound = round(splitEnd);
    midiPitchNumberLookup = notes(k,3);
    stringLetter = notes(k,2);
    filenamePitch = midiPitchMap(midiPitchNumberLookup);
```


Figure 1 - MIDI Splitting Code

```
if stringLetter == 5
        stringName = 'Vibrato_eString_';
    elseif stringLetter == 4
        stringName = 'Vibrato_Astring_';

    elseif  stringLetter == 3
        stringName = 'Vibrato_Dstring_';
```

Figure 2 – Naming Strings

For the most part this was very successful however certain problems were encountered. The original MIDI files worked on contained lot of duplicate information that was generated when recording the files caused by the finger sliding over the string and picks resting on other strings, these are unavoidable when playing guitar however it did mean that the MIDI files had to be cleaned up after their creation to make the information easier to manipulate.

After the files had been initially cleared up some experimentation needed to be undertaken when chopping up certain samples, namely the full picked samples. The problem still being encountered was that the start time of the note as given by the "midiInfo" script was not exactly accurate and there would often be a number of seconds difference between where the note actually started and where the MIDI file said it would. This meant that there would either be a lengthy silence, or the end of the previous note could be heard at the start of a new note. To get around this problem a number of tests were undertaken to add or subtract a certain amount from the start and end value given by the MIDI information. Due to the fact that the notes were not quantised perfectly in the original wav file it was necessary to leave a certain amount of space so that the start of any notes would not be chopped off. I believe the cause of this problem is that the MIDI start time for the note was actually representing when the player first rests their finger on the fret in question, rather than when they pick it.

The time that it took to slice these MIDI files became an issue at certain points due to the fact that the full picked versions of the audio files were very large due to the fact that they were recorded in 10 channel wav format. The smaller files of damp picked notes and the vibrato notes were of a much more manageable size, kilobytes per sample, and so it was possible to process and chop up around 1000 samples in a number of hours. There was however a problem with dividing up the larger sound files, due to their size (the whole wav files containing all the notes was around 400mb) processing and dividing these files became very time consuming, taking around 150 hours to process, during this time the computer that the research was being conducted on was also virtually unusable for any other task and it was, of course, impossible to use MATLAB during this time, this ate into the already limited timescale for this project.

7.3 Data Preparation/Organisation:

It has previously been mentioned that a large amount of guitar sample data was available from the start of this project, almost 30Gbs worth. Although not all of this data was usable, due to the file extension of the files which were mostly designed for specific plugins, a decent amount of relevant guitar samples were available.

One of the main sources of relevant guitar samples was found in the Indiginus Acoustic Guitar Collection guitar sample pack. Examination of this sample pack showed that the files followed some sort of naming structure and that only around half of the files in the sample pack could be used to aid in classification. By further studying the sample library it became apparent that the samples were named using two letters to represent whether it was a note or chord being picked, letters to represent where it came on the chromatic scale

and two letters to denote whether it was a muted sample, and dampened sample, a slide or a harmonic. By studying this data it was realised that only the standard picked samples would be viable for classification as research into how the distribution of energy in bins would work for slides and chords was not available. Despite much of the data being unusable it was decided that there would be enough data to be used in electric or acoustic guitar type classification.

To solve the problem of awkwardly named samples it was decided that a MATLAB script would be created that would allow the user to search all the wav files in a folder for files whose names contained up to two strings of any length defined by the user. For example it would be possible to search a folder for all files containing the strings "abc" and "def". The script then copies the files from this folder and places them in a folder of the users' choice.

Not only was this method of moving data useful for the initial data it later became incredibly valuable when trying to move the files that were created using the MIDI manipulation technique that were incorporated. It was decided that classifiers for each string, note and guitar type and vibrato would be incorporated; because of this a huge amount of file manipulation was required. Due to the nature of how the text files for classification were created it was necessary to have all of the files of a certain class in the same folder. In the case of single note classification this would have taken an extremely long time without the script for moving files as almost 50 different folders were created with 5-25 files being placed in each folder. It is worth mentioning that in previous projects a similar idea was conceived whereby the file name of certain files would be manipulated using regular expressions in the Ruby programming language (Cohen, 2014). It was decided that a more effective method that could easily be used by later, similar project should be made in MATLAB. Fortunately MATLAB provides excellent features for cutting and copying files from folders.

For the file organisation scripts it was decided that there should be a way of both copying and removing files from folders. The thought being that if possible it would be advisable to keep files in their original folders, especially in terms of the data that was present at the start of the project as many of the naming conventions were not unique enough to identify their original folder. By keeping the data in its original folder it means that this data can be easily accessed for future classification attempts. The ability to remove files from folders became very important later on in the project due to issues that occurred during the creation of the wav files from the MIDI manipulation system. Due to teething problems in the MIDI farming scripts which before edited caused duplicate notes it was occasionally necessary to remove files from a folder rather than running the whole script again, considering the 100 hours or more runtime of some of the scripts.

The creation of scripts to move files effectively and quickly became increasingly important as the project progressed. Some classification types returned very low success rates for classifiers where they were expected to be successful and it therefore became necessary to test more specific data sets, and conduct the test enough times with similar data sets to achieve accurate results. Because of this a huge amount of time was spent moving and copying files into certain directories so that they could easily be classified. Without this simple script it would not have been possible to create much of the data that has been produced.

7.4 Inharmonic comb filter feature extraction:

The key piece of code that allowed this project to proceed was the extraction of inharmonic comb filter feature vectors originally defined in "Signal representation and estimation of spectral parameter by inharmonic comb filters with application to the piano" (Alexander

Galembo, 1999) this was used to predict the fundamental frequency on piano bass strings. The inharmonic comb filter (ICF) allowed for the extraction of harmonics only from the FFT (fast Fourier transform), producing a feature vector of at most 30 elements long to work with, this is much simpler than dealing with a feature vector that can potentially be thousands of elements long. The ICF MATLAB code was originally produced by Pritchard in his Masters Thesis (Pritchard, 2002). This code was then further modified by James Cohen (Cohen, 2014) in a previous related project. By using the ICF developed by Pritchard it is possible to extract:

- A feature vector containing up to the first 30 harmonics
- Fundamental frequency prediction
- Inharmonicity estimation.

Pritchard's implementation of the ICF works by computing the FFT of an audio file and then looping over the FFT three times detecting bands where energies are stored. This is then used to compute an accurate fundamental frequency estimate. When using the ICF the user is required to enter the audio file name, an estimate inharmonicity value and an offset percentage. In "Signal representation and estimation of spectral parameters by inharmonic comb filters with application to the piano" (Alexander Galembo, 1999) an average value for inharmonicity is defined, typically a value of 0.003, which is also outlined in the paper, you can see the equation for this below:

$$B = \frac{\pi^3 D^4 E}{64 T L^2}$$

Equation 1 – Inharmonicity

Certain modifications to Prichards ICF were later made by Cohen during his "Machine translation of guitar audio" project. These changes included:

- Changing the range of the fundamental frequency accepted from 20-200 Hz to 20-1500Hz to accommodate the larger range of guitar notes that would need to be classified.
- Code to decrease the bin size of the ICF as the original code would attempt to access values form the FFT that were out of bounds for higher frequency guitar samples.

7.5 Audio Analysis:

During a previous project by Cohen it was shown that a certain level of difference seen in different guitar audio samples could be used for classification. During the first stages of this project similar steps were repeated to see if any difference in the feature vector could be seen. It was possible to see, by looking at the feature vectors, that the numbers were not identical, however by using MATLABs 3D bar chart functions it was easy to plot these feature vectors next to each other to visualise how the classifier could be used to identify the differences between feature vectors. Below is an example of a number of random guitar samples that have been run through the ICF and have had their feature vectors plotted (Figure 3). It can be seen that there is a noticeable difference in the feature vectors.

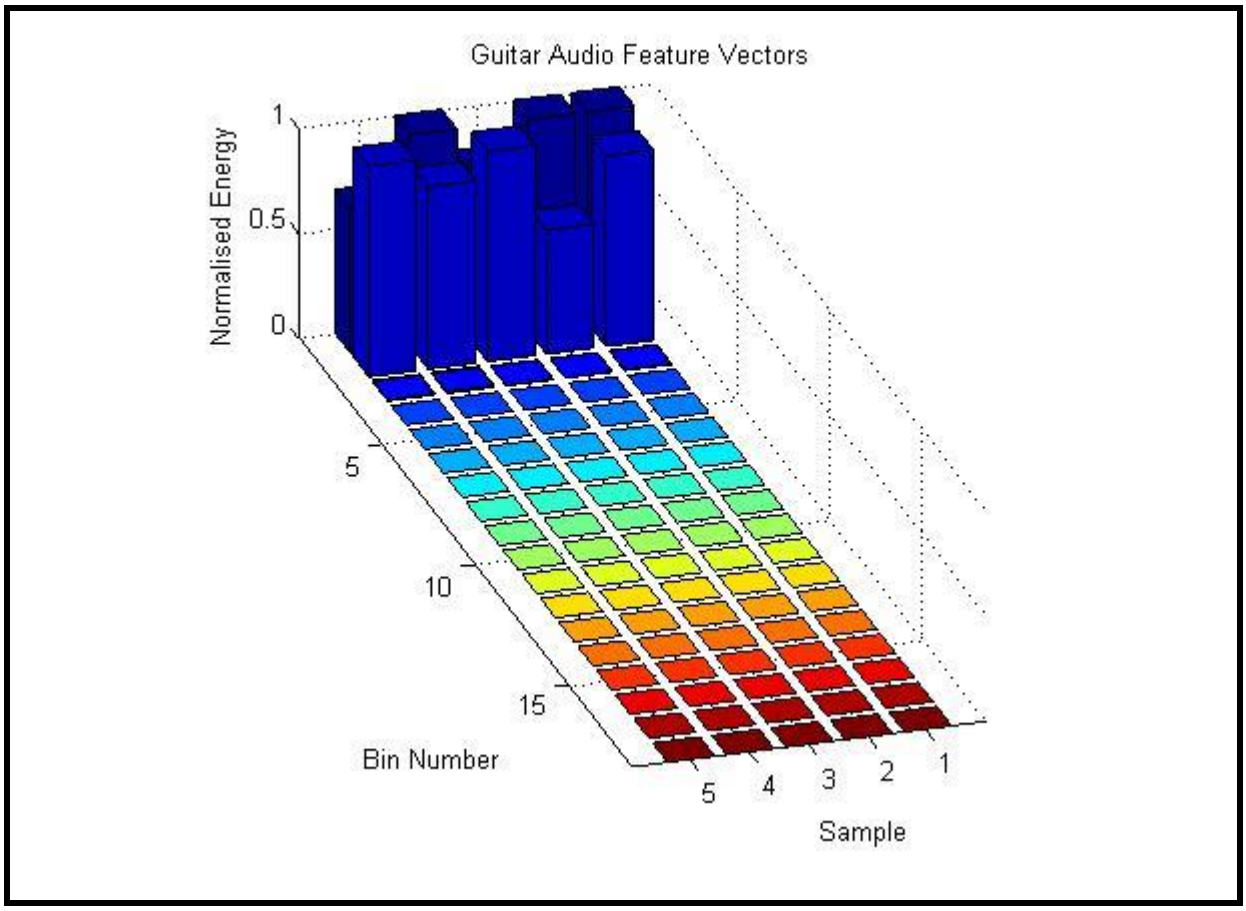

Figure 3 - Selection of Random Feature Vectors

It was important to reaffirm the work previously done by Cohen as it was not noted which guitar samples were using during that project. It has however been mentioned in Cohen's report that only audio samples personally recorded were used for certain types of classification.

As the idea of using the Boss-GP10 to farm audio files evolved it was decided that these emulated samples should also be tested to see if they displayed the same characteristics as recordings of real guitars. It can be seen below (Figure 4) that the farmed data displayed similar characteristics to that of the real guitar recordings. it was therefore decided that this data could be used for the creation of classifiers.

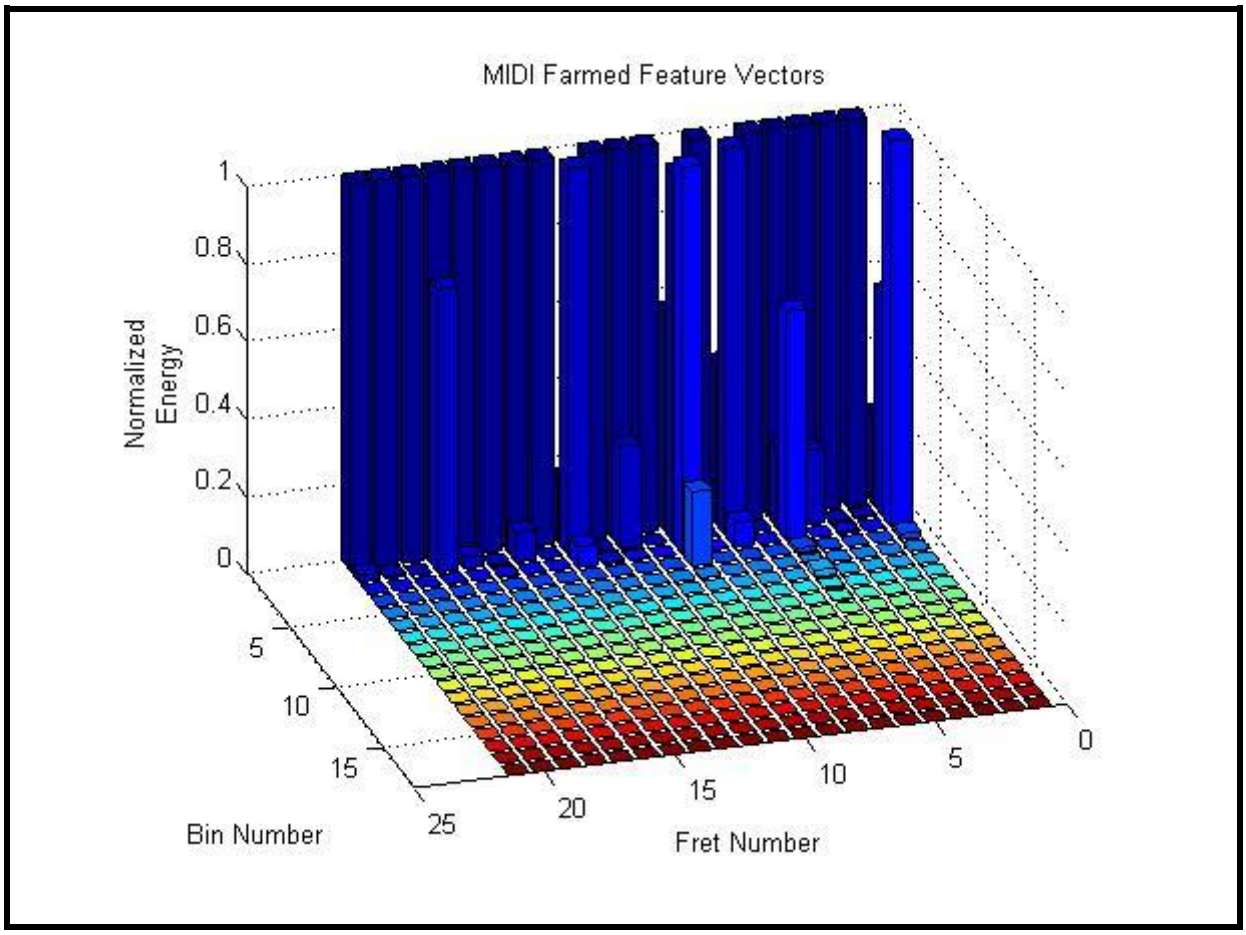

Figure 4 - MIDI Farmed Feature Vectors

7.6 Normalizing feature vectors:

This was an essential part of the project and was vital to insuring that the results were accurate. The theory behind normalizing the feature vectors is that exact conditions in which the guitar samples have been recorded are unknown, and it can be assumed that even notes that are played at identical intensities could seem louder or quieter due to differences in recording conditions. The best way to normalize the feature vectors is to find the maximum power value of an array and divide all the other elements by that value.

7.7 Machine Learning Classification:

This was the most vital part of the project. The idea of classification is that if you create a number of classes, and feed a large amount of data into each class that will then come to define that class. By comparing the information requiring classification with data in the classes, it should be possible to define that data by the class that it most closely resembles.

During this project the feature vectors extracted from the audio files using ICF were used to build classifiers. It is possible to build classifiers with large numbers of classes within using LibSVM and this includes a tool that allows the user to easily write the classes in a LibSVM readable format. LibSVM also provides an exceptionally easy way of reading text files containing pertinent data and class prediction methods.

During Cohen's project it had been proven that classifiers could be used to determine between acoustic and electric guitars, as well as plucked intensity. Due to the many guitar types that became available for classification after farming data from the Boss-GP10 more specific classification of guitar types was investigated. It has previously been mentioned

that 12 String, Telecaster, Stratocaster, Martin, Sitar, Banjo and Les Paul emulations were harvested from the Boss-GP10, each with full and dampened picked versions of 21 notes on each string. As such a classifier with all these guitar types was investigated.

Other tests using confusion matrix and receiving operator characteristic curves (ROC Curves) were also used to analyse the information gained from classification attempts. The reason for using these techniques is that allows the user to see which classes are being correctly classified and which classes are achieving low rates of classification success. These tests provided immense insight into how effective classification attempts were and were used throughout this project to make informed decisions.

7.7.1 Guitar Type Classifiers:

Based on the successes that had occurred during Cohen's project it was originally assumed that the classification of these different guitar types would have a very high success rate. However upon testing all of the guitar types against each other in a seven class classifier success rates were low (37%), due to having a larger number of classes and having higher similarities between certain guitar types than between electric and acoustic guitar types. Due to these unexpected results more specific testing of certain types of guitar against other types of guitar in two class classifiers was undertaken to see what was causing the misclassification of samples.

After further investigation a classifier was devised that only contained the damped versions of the notes, and another that only contained the full picked version of the notes. The results from this test were insightful as the damped versions of the notes only returned a 35% success rate, whereas the non-damped versions returned an 80% success rate. After this low classification rate, the feature vectors were analysed on the 3D bar chart. Below can be seen the same note in both damped and full pick variation (Figure 5), where the damped note is 1, and the full picked note is 2. It can be seen that the damped version predictably showed less energy as the bin number increased. Based on the consistency of this information it was concluded that the damped picked versions of all the notes loose enough of the defining quality that makes them classifiable by the damping of the note as it is played for them to not be used in later classification efforts, however further classification attempts involving the full picked and damp picked versions of the notes were carried out separately and together.

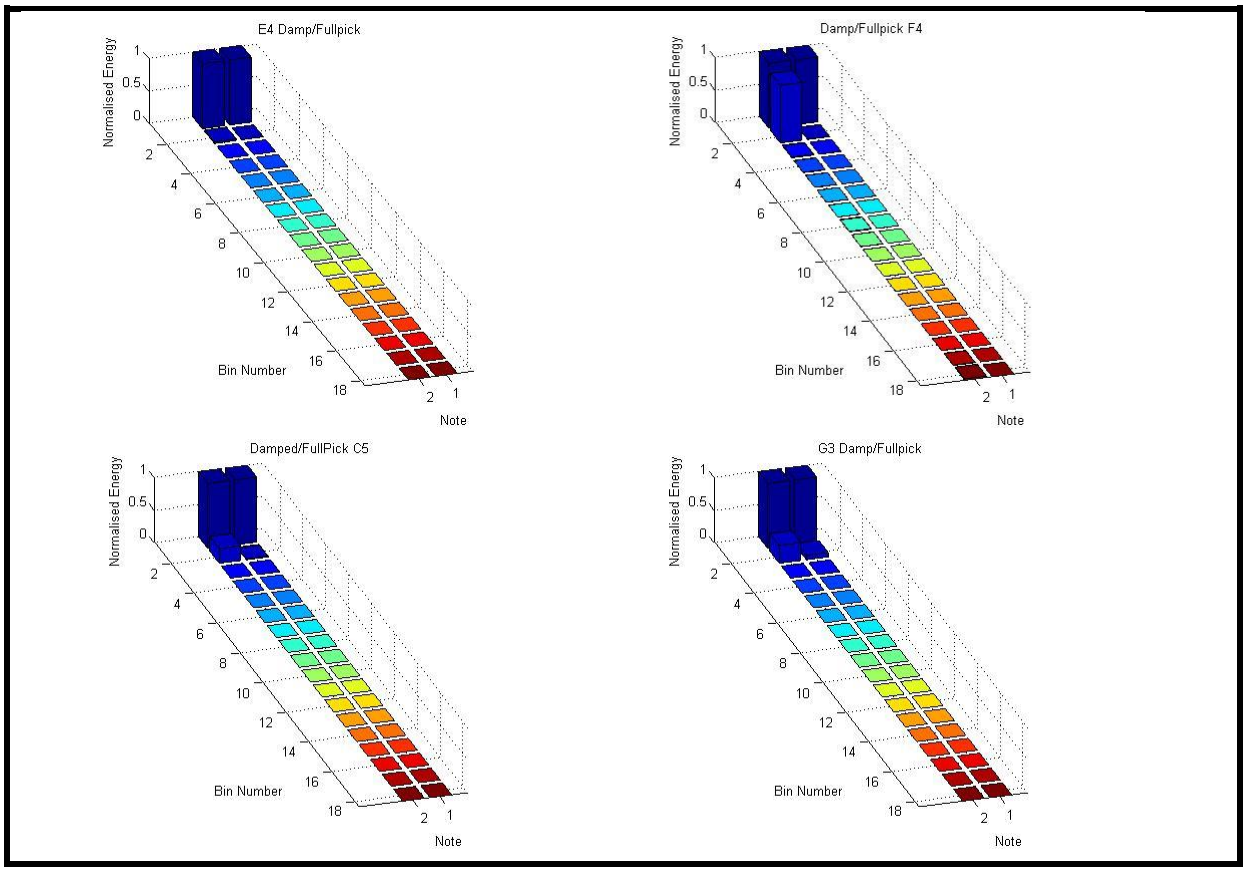

Figure 5 - Damp Picked VS Full Picked Notes

7.7.2 Individual Note Classification:

The original testing of the single note classifier was done with both the damp and full picked versions of the notes. This however returned an extremely low classification success rate (6%). Classification without damp picked versions of the notes was also undertaken after it was realised that this could affect the success rate of the classifiers, success rate did improve however it was still very low (10%). There were over 40 different classes involved in this classifier due to the number of notes on a guitars neck so the results hinted that more specific classifiers of the some nature could be created. By looking at graphs displaying the total energy in a range of notes in can be seen that there is a discernible difference between them.

In theory the difference in notes should have been enough to differentiate between the different notes; however this difference was not high enough between a wide range of notes to give each note enough individuality to be classified. Below is an example of a number of frets on different strings and the spread of their energy in different bins (Figure 6). As it can be seen, although there is significant difference between some of the notes, many others are almost identical.

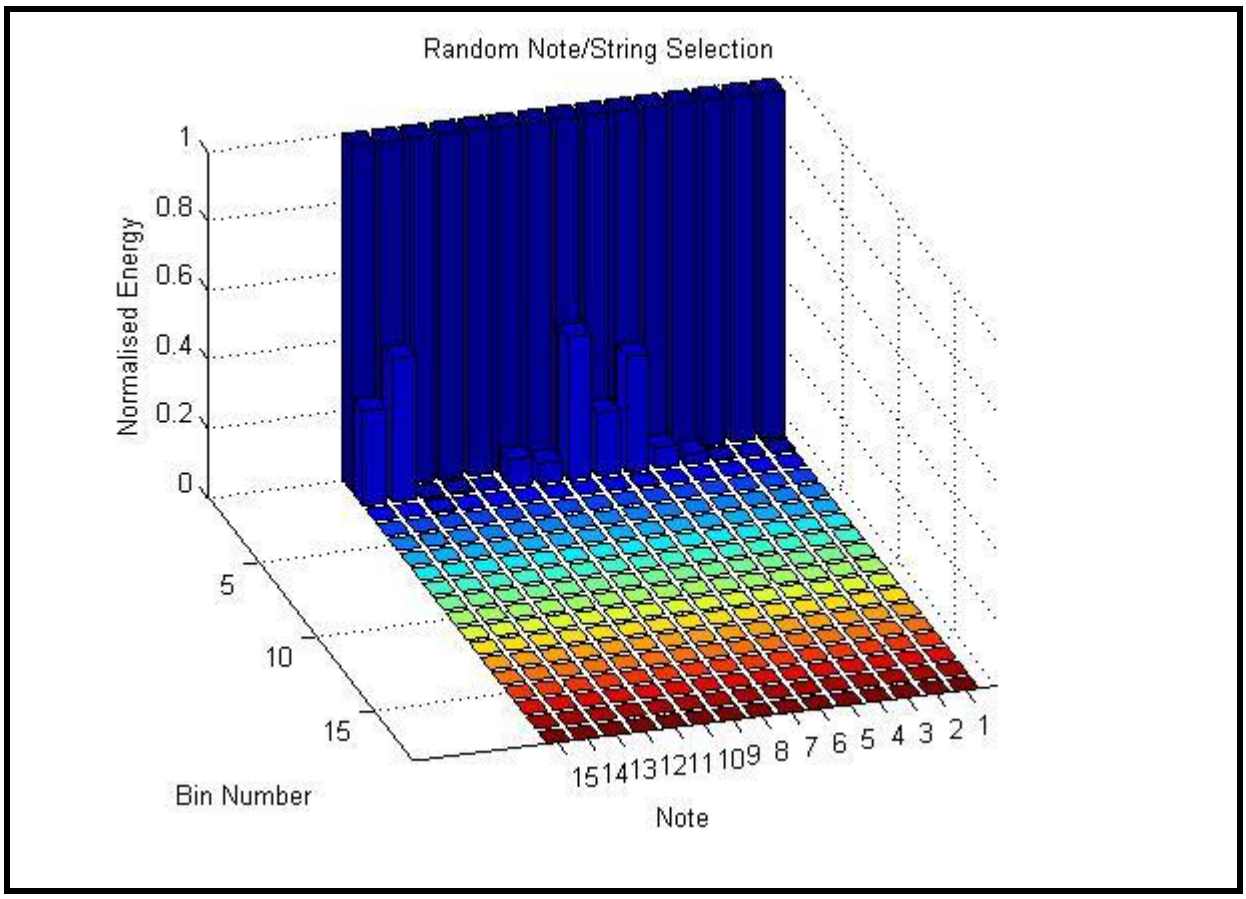

Figure 6 - Random Notes, Random Strings Feature Vectors

I concluded that this is the reason that the creation of a multiclass classifier for all of the different notes on the guitar's neck was not possible.With the current data set this testing was largely superficial as it is possible to gain the fundamental frequency accurately from a note by using the ICF, however the ICF can sometimes be inaccurate to a few degrees so it was worth testing to see if this could be a more accurate way of predicting notes. It did give a better insight into the capabilities of the classifier. Single note classification for a wide range of notes was also undertaken for each string; investigating whether by only involving notes from one string in the classifier the accuracy rate could be improved upon. The results were a slight improvement however not enough to be considered a more effective method of exact pitch detection than the ICF. By looking at the classification matrix for a classifier built with all the notes on all the strings (Appendix F) you can see very little correlation however the results are not entirely erratic, based on this confusion matrix it may be possible with a much larger data set to build an exact note prediction mechanism based purely on machine learning.

7.7.3 Note position Classifiers:

After it was proven that it would not be possible to simply machine learn all of the notes it was decided that a better way of finding the position of the note would be to gain the fundamental frequency from the ICF and then use classifiers to determine which string the note was being played on, as the same note can potentially be played on five different strings on a 21 fret guitar. Some small bar charts were created with the same note on the different strings to see if there would be a visible difference in the feature vectors to decide if it would be worth continuing with this method.

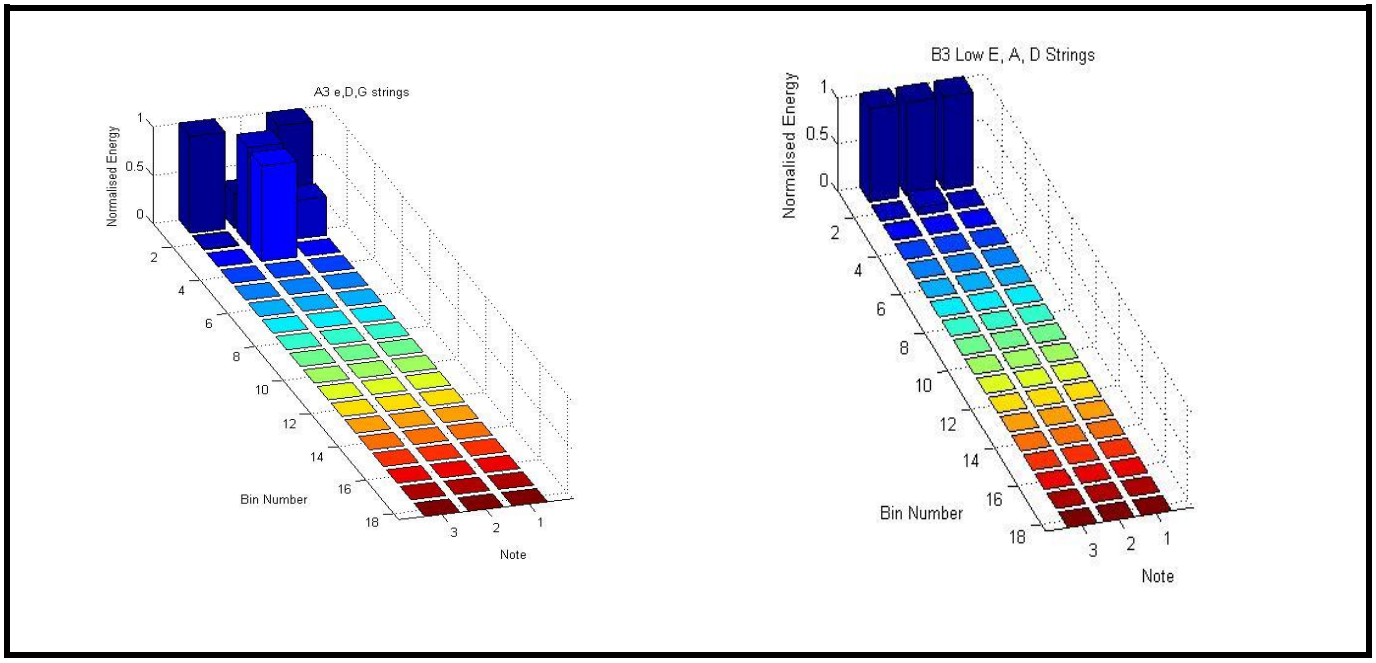

Figure 7 - Same Note Different Strings

In the bar charts above you can see a clear difference in the feature vectors between the same note on different strings. These charts were created using the full picked versions of notes from the Les Paul emulation. Although the differences in the feature vectors for B3 were not as prominent as those in the A3 feature vectors the results were promising enough to continue with this kind of classification.

7.7.4 Vibrato Classifiers:

Classifiers were also built to test vibrato. As in the previous tests the feature vectors were analysed to see if there was enough difference between them to create a successful classifier. The purpose of this classifier was simply to test a range of notes without vibrato against a range of notes with vibrato and see how accurate classification could be. Testing with single notes with both vibrato and non-vibrato was done and testing with all available notes containing vibrato against all full picked variations of the notes that did not contain vibrato was undertaken. For this classifier all of the guitar types were tested individually to assess whether the tone of some of the guitars may unintentionally create a vibrato style effect without vibrato actually being applied to the note

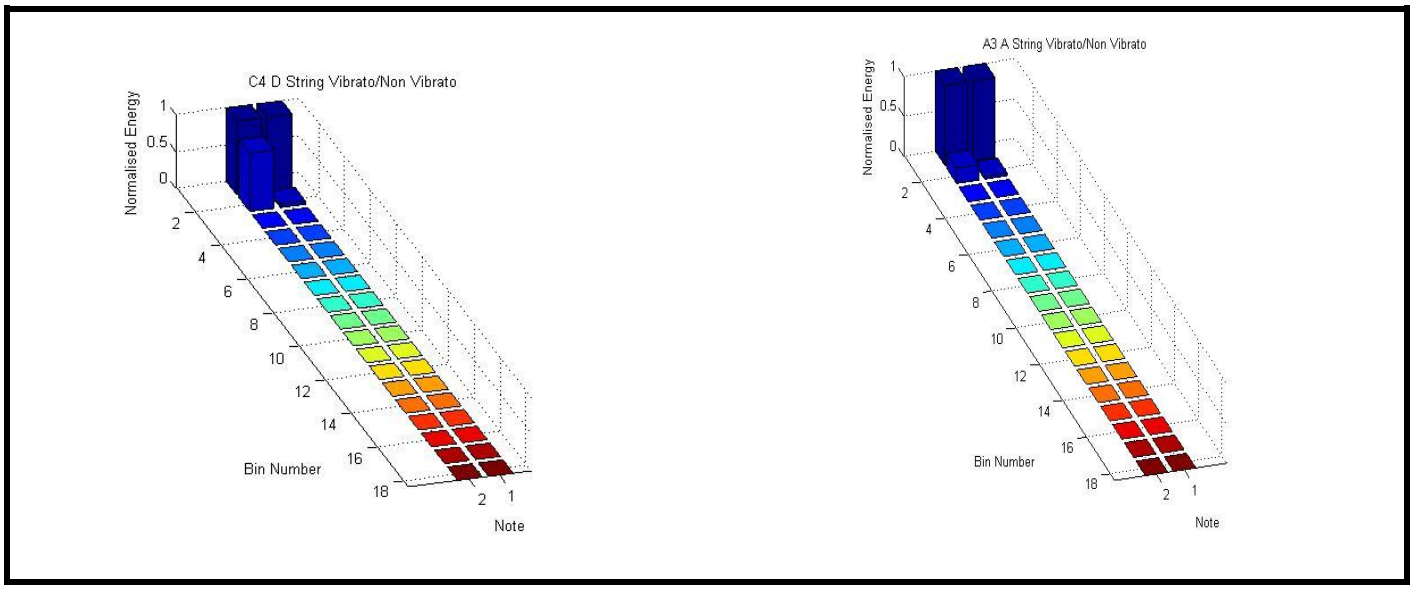

Figure 8 - Vibrato VS Non-Vibrato

The figures above (Figure 8) show vibrato and non-vibrato comparisons of notes A3 and C4, the first note is the non-vibrato note, and the second the note with vibrato played on Les Paul guitar emulations. It is clearly visible that on both occasions the version of the note with vibrato contained more energy. This was considered sufficient enough evidence that some form of vibrato classification could occur.

7.7.5 String Classifier:

A classifier for identifying which string the note was being played on was also undertaken. It was expected that due to the different thicknesses of the string that the notes are being played on there would be enough of a difference for the classifier to pick up. However initial testing of this containing both the damp and full picked versions of the notes returned an unexpectedly low success rate (27%) . Further testing without the damp picked notes returned a classification success rate of 35%. This was a slight improvement however it was still not enough to be used in an effective method of transcription.

7.7.6 Similar Guitar Type Classifiers:

As of some of the instruments that were emulated on GP-10 were not guitars, the banjo and sitar for example, further tests were conducted between the Sitar, Banjo, 12 String and acoustic guitars. The reason for conducting these tests was that Sitars, 12 Strings and Banjos are also technically acoustic instruments and although differences between the audio samples for each instrument can be heard it was initially thought that this could have been the reason for the low classification success rate.

Further guitar type testing was done between the Strat, Tele and Les Paul electric guitar types to see if the differences between them could be utilized to build a successful classifier, this lead to some interesting results which seemed to suggest that this would be possible, The reason these guitars were tested against each other was because they are all emulations of electric guitars, where differences in feature vectors were likely to be less than

differences in feature vectors between electric and acoustic guitar types as the audible difference is much smaller.

After building and testing a range of classifiers it was decided that classifiers should be built for the same note being played on different strings. In certain cases it is possible for the same note to be played on three different strings if all 21 frets of the guitar are included. The theory being that it would then be possible to find the fundamental frequency using the ICF prediction method and then use the classifier to see which string the note is being played on.

8. Results

Presented Here is a collection of all the results for the different classifiers that I have previously outlined, in the appendix there are also ROC Curves and Confusion Matrices for all the results apart from the same note played in a different position results, as it was felt creating ROC Curves and Confusion Matrices for these results would not provide any further insight. The purpose of undertaking these tests was to determine whether the method of classification using feature vectors extracted from audio files, which contain a recording of a single note on the guitar neck being played, was possible. If it was possible to correctly predict the position of the note being played through the use of a classifier, or by other methods using a combination of classification and feature vector manipulation, this method could then be implemented to create a functioning system.

8.1 Guitar Type Classifiers

Displayed below are the results of the tests for guitar type classification. A wide range of tests were conducted on guitar type in an attempt to emulate the success achieved by Cohen in his original project (80%). The initial Acoustic or Electric classifier created produced very similar results using data that was not farmed from the Boss-GP10. The reason for the extensive guitar type testing was that when the MIDI farmed data was introduced classification rates became much lower.

8.1.1 Collected Guitar Type Classifier Testing

Classification Type	Number of files used for training	Number of files used for testing	Classification Accuracy
Acoustic/Electric	396	99	75.757% (75/99)
Acoustic/Electric/Other Full Picked	1196	229	68.896%(206/299)
All Guitar Types All Note Styles	1376	344	37.209%(128/334)
Guitar Type Damp Picked	704	176	35.795%(63/176)
Guitar Type Full Picked	672	168	32.142% (54/168)

Table 1 - Collected Guitar Type Classification Results

From the above results (Table 1) it can be seen that successful classification occurred for Acoustic or Electric guitar types and similar rates of success occur when a new type named "Other" is introduced, where other represents something outside of the non-conventional guitar type such a Banjo or a Sitar. Due to the low classification rates that occurred for both damp and full picked classifiers containing all the guitar types, classifiers comparing two guitar types were created. By looking at the confusion matrices for the classifiers (Appendix A) containing all guitar types it can be seen that although the classification accuracy was low for a classifier with many guitar types, there were certain guitar types such as the 12 String and the Les Paul that achieved a very high classification Accuracy.

8.1.2 Single Guitar Type Classifier Testing

Classification Type	Number of files used for training	Number of files used for testing	Classification Accuracy
Strat against Les Paul	196	48	52% (25/48)
Strat against Tele	400	100	62 % (62/100)
Tele against Les Paul	196	48	64% (31/48)
Sitar against Martin	400	100	69% (69/100)
Sitar against Banjo	400	100	65% (65/100)
Sitar against 12 String	400	100	77% (77/100)
12 String against Banjo	400	100	65% (65/100)
12 String against Martin	400	100	68% (68/100)
Banjo against Martin	400	100	65%(65/100)

Table 2 - Single Guitar Type Classification Results

Above can be seen very high rates of classification success when individual guitar types are tested against each other (Table 2). By looking at the high rates of success achieved by comparing two distinct guitars, and the success rates gained when comparing Electric, Acoustic and Other a conclusion can be drawn that a combination of these classifiers could be used to create a very accurate method of guitar type detection, with the accuracy increasing as more data becomes available to aid with classification. By looking at this data and the confusion matrices for this table (Appendix B) you can clearly see that certain electric guitar types such as the Les Paul have a very high rate of classification, whereas the Strat tends to have a much lower rate.

8.2 Expressive Techniques

8.2.1 Vibrato Testing

Classification Type	Number of files used for training	Number of files used for testing	Classification Accuracy
Combined Vibrato/Non-vibrato	2485	352/497	70%(352/497)
Martin Vibrato/Non-vibrato	250	50	64% (32/50)
Les Paul Vibrato/Non-vibrato	250	50	60% (30/50)
Tele Vibrato/Non-vibrato	250	50	54% (27/50)
Strat Vibrato/Non-	250	50	56% (28/50)

vibrato			
Banjo Vibrato/Non-vibrato	250	50	56% (28/50)
Sitar Vibrato/Non-vibrato	250	50	40% (20/50)
12 String Vibrato/Non-vibrato	250	50	60% (34/50)

Table 3 - Vibrato Classification Results

The above results (Table 3) make it apparent that a classifier for expressive techniques such as vibrato could be created by using classifiers. Reasonably high rates of classification occur for most of the guitar types. In the case of the Sitar a lower classification occurred. By looking at a bar chart displaying the feature vectors of notes on the Sitar with and without vibrato it can be seen that there is a certain amount of difference, however this is not as prominent as the differences between notes with and without vibrato on other guitar types. Below there are comparisons of non-vibrato and vibrato for Martin emulation and the Sitar emulation (figure 9).

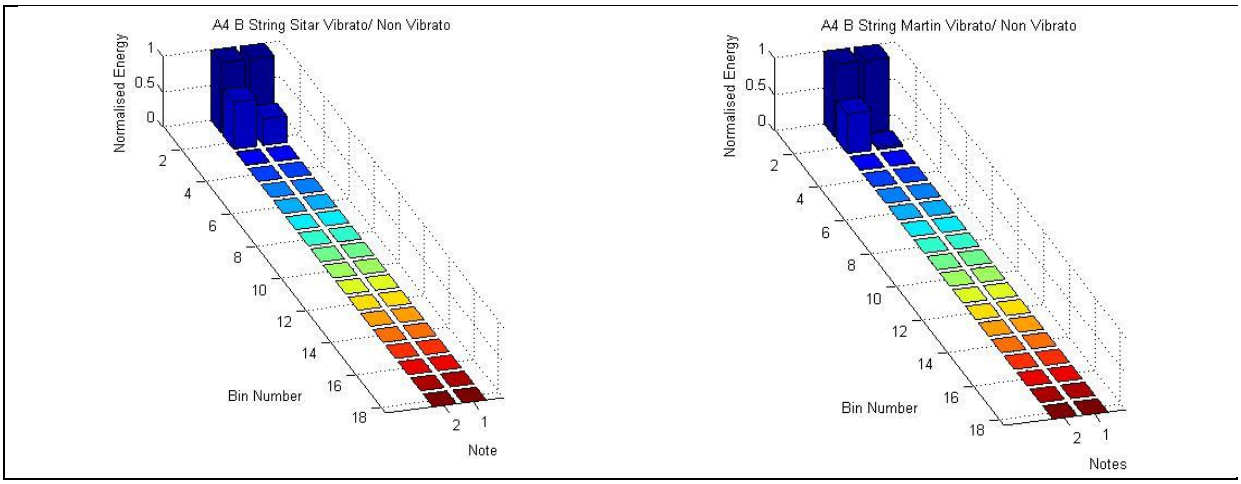

Figure 9 - Sitar Vibrato VS Non-Vibrato

By listening to the audio of the notes represented in the above figures it can be heard that the Sitar naturally has a vibrato like ringing to its tone. If more data can be collected I don't think that the issues outlined would cause a significant enough problem to stop an effective classification method being implemented. The other results certainly indicate that this type of classification would be possible for the majority of guitar types tested during this project.

8.3 Same Note Played on Different String, Full pick Note

Below the results of extensive testing of note detection using the classification method can be seen (Table 1-8). A large amount of testing was required due to poor results being achieved from the classifier which contained all the notes on the neck, as well as the classifier that compared the same notes on the same string. A further reason for the range of notes being used is that these tests give a good insight into how the position on the neck in which the note is being played affects the amount of energy in the different bins. Damp picked versions of the notes were not included in this test as previous tests involving damp picked notes returned poor classification results. As such they were tested separately.

Low E/A String

Note/Octave	Number of files used for training	Number of files used for testing	Classification Accuracy
A2	24	6	100% (6/6)
A#2	24	6	83.333% (5/6)
B2	24	6	83.333% (5/6)
C3	24	6	66.666% (4/6)
D3	24	6	66.666% (4/6)

Table 4 - Low E/A String Classification Results

A/D String

Note/Octave	Number of files used for training	Number of files used for testing	Classification Accuracy
D3	24	6	100% (6/6)
E3	24	6	50% (3/6)
F3	24	6	66.6% (4/6)
C4	24	6	83.333% (5/6)
E4	24	6	50% (3/6)

Table 5 - A/D String Classification Results

D/G String

Note/Octave	Number of files used for training	Number of files used for testing	Classification Accuracy
G3	24	6	100% (6/6)
A3	24	6	100% (6/6)
B3	24	6	100% (6/6)
G4	24	6	100% (6/6)
F4	24	6	100% (6/6)

Table 6 - D/G String Classification Results

G/B String

Note/Octave	Number of files used for training	Number of files used for testing	Classification Accuracy
B3	24	6	100% (6/6)
B4	16	4	100% (4/4)
C4	24	6	50% (3/6)
C5	24	6	33.333% (2/6)
D4	24	6	50% (3/6)

Table 7 - Low G/B String Classification Results

B/High E String

Note/Octave	Number of files used for training	Number of files used for testing	Classification Accuracy
E4	24	6	83.333% (5/6)
E5	24	6	83.333% (5/6)
F4	24	6	66.666% (4/6)
F5	24	6	50% (3/6)
A4	24	6	50% (3/6)

Table 8 - B/High E String Classification Results

By reviewing the above results it can be seen that this method of classification has proved successful. Even with a very limited data set many tests resulted in a 100% classification success rate. The average success rate of these tests was 76.48% which is very positive, considering that there are a number of 33.333% results which when compared to the amount of other positive results are likely to be caused by issues with the data, rather than a fundamental flaw in the theory. It has previously been noted that due to the method of data farming used, some audio files had a tiny bit of the last note played at the start of them. This could change the amount of energy in some of the feature vectors causing classification to be less accurate.

8.4 Same Note Played on Different String, Damp pick Note

Note/Octave	Strings	Number of files used for training	Number of files used for testing	Classification Accuracy
B3	Low E, A, D,G,	20	4	75% (3/4)
A#3	Low E, A, D, G	20	4	50% (2/4)
C4	A, D, G, B	20	4	50% (2/4)
F4	E, D, G, B	20	4	80% (4/5)
C5	High E, B, G	15	3	33.333% (1/3)
G5	High E, B, G	10	2	50% (1/2)
G#5	High E, B, G	10	2	100% (2/2)

Table 9 - Same Note Different String Damp Picked Classification Results

The table above results of the same note in damp picked variety being played on a number of strings (Table 9). As has constantly been a problem throughout this project the amount of available data has cast a shadow over the accuracy of the classification results. As with the full picked note testing across various strings promising results can be seen. As expected these results are not as promising as the full picked versions of the notes, an issue that has been apparent in nearly all the tests conducted. The cause of this as outlined previously is likely to be due to the fact that there is less total energy in the feature vectors, causing the differences between the notes to be much smaller. These results, alongside those of the similar tests conducted with the full picked notes show that this can be used as an effective

method of string detection. By looking at the confusion matrices and ROC curves of these tests it can be seen that the drop in classification accuracy is most commonly down to the note being classified as one sting above or below the string the note is actually being played on, rather than a difference on two or three strings. This leads to the conclusion that with the introduction of more data these sorts of results would begin to disappear.

8.5 Same String Different Note

Below are the results for classification of notes on the same string (Table 10). It becomes immediately apparent that this method of classification was less successful than many other classification attempts. For each string two notes have been tested against each other, always in close proximity on the fret board to see if classification can occur. Outlined in red is also the classification of all notes on the string for the first 12 frets, again these results are not as promising as one could have hoped however considering the amount of classes the classifier involved they are not entirely negative.

String	Note/Octave 1	Note/Octave 2	Number of files used for training	Number of files used for testing	Classification Accuracy
Low E	A2	A#2	30	6	33.333% (2/6)
Low E	C3	D3	30	6	50% (3/6)
Low E	Frets 1-12	Frets 1-12	345	69	24.637% (17/69)
A	C3	B3	30	6	83.333 (5/6)
A	A2	A#2	30	6	66.666% (4/6)
A	Frets 1-12	Frets 1-12	195	39	35.897% (14/39)
D	A3	B3	30	6	83.333%
D	F4	G4	30	6	50% (3/6)
D	Frets 1-12	Frets 1-12	190	38	7.894% (3/38)
G	B3	C4	30	6	50% (3/6)
G	B4	C5	30	6	50% (3/6)
G	Frets 1-12	Frets 1-12	195	39	17.948% (7/39)
B	C4	D4	30	6	83.33% (5/6)
B	B3	C4	30	6	33.333% (2/6)
B	Frets 1-12	Frets 1-12	175	35	25.714% (9/35)
High E	E4	F4	30	6	83.333% (5/6)
High E	F4	G4	30	6	50% (3/6)
High E	Frets 1-12	Frets 1-12		36	17.647% (6/36)

Table 10 - Same String Different Note Classification Results

By looking at the confusion matrices of the results in Table 10, it can be seen that certain strings present promising signs of effective classification of all notes on one string, for the example the B String (Appendix E5) however some strings such as the G String (Appendix E4) show far less promising results. With the introduction of more data it may be possible to decide whether or not this could be an effective classifier however currently it cannot be said if it will or will not work.

9. Conclusion

By looking at the results gained from the testing conducted during this project I concluded that the most effective method of creating a system that has the ability to not only transpose music into guitar tablature, but also detect what type of guitar is being played as well as other features such as expressive techniques that are being applied to the note and length of the note that is being played would be an amalgamation of many of the classifiers that I have created.

The most logical option would be to use the inharmonic comb filter created by Prichard (Pritchard, 2002) to gain the fundamental frequency of the note. From this note it would be possible to classify where on the neck and upon which string the note is being played by utilizing a classifier containing all the different possible options of where that note can be played. This note could also be passed through the Electric Acoustic or Other classifier to identify on which type of guitar it is being played. Once this is determined it would then be possible to find more specific information about what type of, for example electric guitar, the note is being played on.

By looking at the confusion matrices and ROC curves creating using the information from 12 frets on the same string classification attempts it can be seen that it may be possible to create an accurate method of transcription based purely on machine learning techniques, where all the notes on the string are put into an individual class. For this to be tested any further much more data will be needed, however the fact that my testing has not disproved it means that it should be investigated further.

By looking at the results presented it can be seen that classification of the same note on different strings returns a very high success rate. String detection was one of the most important aspects of this project due to the inharmonic comb filter's pre-existing ability to predict frequency. I believe that the success of this classifier alone is enough the warrant further research into this subject. The level of success that has been achieved in other classifiers also presents the possibility of a much wider range of guitar type classifiers that is only limited by the amount of data available. Other tools developed during this project such as the wav slicing tool need further development however they present a reasonable grounding upon which a truly effective system for transcription could be created.

During this project results have been gained from extensive testing of theory outlined in "Signal representation and estimation of spectral parameter by inharmonic comb filters with application to the piano" (Alexander Galembo, 1999). By utilizing a method of extracting feature vectors devised by Pritchard in his Masters Thesis (Pritchard, 2002) and building upon research done by James Cohen in his Machine Translation of Guitar Audio (Cohen, 2014) I feel I have managed to conclusively prove that this can be used successfully to transpose guitar audio successfully, as well as provide a set of tools that can be used to generate more data to create more reliable classifiers.

10. Future Work

During this project I have created a set of tools and results that have helped prove that this method can be used for guitar transcription. Upon the completion of this project I am satisfied that I have made significant progress on the development of a fully functioning system however there is still further work to be done.

10.1 Inharmonic Comb Filter Adjustments:

One theory for the lack of success in the classification of the damp picked versions of the notes is that the bin sizes in the ICF were incorrectly sized to deal with these kinds of picked notes. Unfortunately there was a lack of time available during this project to investigate this issue further however the small amount of successful classification achieved points to the fact that with potential adjustments to the ICF and more data, a better and more accurate classifier could be created.

10.2 Data Farming:

Lack of proper data has been the Achilles heel of this project and other related projects. During this project I have come up with a method to effectively utilized MIDI files to farm a wide range of samples. Although it was possible to farm far more samples than were previously available to me this system should only be limited by the amount of data that can be obtained.

10.3 Audio Slicing:

The audio slicing tool that I have developed is currently very rudimentary, although it allows for the pitch prediction of individual notes in a large audio file it does have a number of problems that need to be addressed. The first of these is correctly outputting the number of notes that occur in the full wav file. Because of the methods way of using over lapping windows to make sure that all the notes are captured, there are often duplicates of one note analysed. This is a difficult problem to solve as in a piece of guitar music someone could potentially hold a note for 3 seconds, and then strum the same note 6 times in a second. This tool does not have the capability of detecting the number of notes predicted to that degree, it main strengths lie in the prediction of notes that are changing in pitch.

One method that could be implemented is a form of beat slicing that takes a Fourier transform of a segment of the audio file, and then uses a windowing method to go through the frequency information and determine when high levels of frequency occur, based on the assumption that it will be possible to detect the point at which a note is struck due to a frequency spike. Currently the method I have presented of slicing audio files which are then run through the inharmonic comb filter raises another problem in that an average pitch of a number of audio files is taken, however if you wished to then pass this averaged frequency estimate into a classifier to find where on the neck the file is being played then the audio file would need to be recreated. Using the Fourier beat detection method it should be possible to accurately detect the start times of notes and eliminate this problem.

It may also be possible to introduce a BPM counter and base the window size and the amount of windows that are averaged on the amount of notes typically played in a bar of guitar music. This could potentially require large amounts research and relevant data may be hard to obtain.

10.4 Expressive Techniques:

Currently the only expressive technique that has been used in classification has been vibrato, however as previously explained it should be possible to create a set of classifiers for slides and bends. This would require the inclusion of more data, however with further investigation into how MIDI portrays bends and hammer-ons it may be possible to manipulate MIDI files to farm data in expressive techniques. I believe it would be possible to create a more reliable plucked intensity classifier than was previously created by Cohen, as well as including slides, bends and hammer-ons into the system, this could even stretch as far as identifying what thickness of pick is being used, or if no pick at all is used.

10.5 Further Guitar type and Tuning Classification:

The results from this project show that even very similar guitar models such as the Tele and the Les Paul models can be classified to some effect, and this should improve with the inclusion of more data. Because of this I have concluded that the number of guitars that can be classified is only limited by the amount of data that can be collected. I further conclude that it should also be possible to include different tunings into a functioning transcription method.

11. Reflection on Learning:

This project was based on an interesting topic and utilized concepts that are relatively new to the world of computing and proved a useful learning exercise. Throughout the course of this project I have learnt and come to understand a plethora of new concepts, some which I had set out to discover more about, and some I had not known I would need to know anything about.

11.1 Organisation:

This project stretched my organisational skills on a number of levels. The original project plan was a useful tool for helping me with my time management however many unforeseen problems occurred slowing the progress of this project. In addition the project never had a definitive end point I therefore had to prioritise looking into certain areas, this needed to be balanced with the requirement to write a good report clearly identifying the results obtained. Project time constraints have meant that certain systems such as the wav slicing tool are not as effective as I would have liked, however I made the decision that further research into classification techniques would be a more beneficial use of the time available to me.

Organisation of data and files was another unexpected issue. As I created more and more classifiers organisation of files on my computer became exceedingly complicated and I began to see how important correct annotation of data and file structures was. This project included periods of immense frustration when I had to continually click through the windows folder hierarchy to find specific audio files or classifiers, and I realised the importance of being able to easily access the required folder and files quickly.

11.2 Programming Ability:

I started this project with a limited ability to program in MATLAB. I had an understanding of the basic techniques however undergoing this project was hugely beneficial for helping me extend my knowledge and skills in an area where I previously considered myself to be weak. During this project I was able to implement some techniques that I have learnt throughout my time at university, for example hash maps and adaptations of binary searches. I found it very rewarding to be able to implement these programming concepts and see them have a positive effect on the results I was trying to achieve.

Using scripts gained from external sources has formed a large part of this project. They have not only allowed the project to succeed but have also increased my understanding of a range of subject. For example though the use of the Ken Schutte MIDI scripts (Schutte, 2012) I gained a broader knowledge of how MIDI works, and how MATLAB can be used to processes MIDI and extract relevant information.

11.3 Digital Signal Processing:

Throughout this project much of the work done to audio files involved the use of digital signal processing techniques and functions built into MATLAB. Much of the work I had previously done on MATLAB had been related to DSP and so it was therefore interesting being able to apply these techniques to a project that I felt an affinity towards.

11.4 Guitar Audio/ Machine Learning:

Having played guitar for many years I leapt at the chance to be able to do my final year project on a subject I was truly passionate about. During the course of this project I have been able to better understand how the construction of notes works and the role that a vibrating string plays in creating sound and how these features can be utilized to determine other information about the piece of guitar music that is being played.

Another subject that I have had little knowledge of prior to this project is machine learning. Having had little prior knowledge of the subject I took the opportunity to learn as much as I could about the subject. Not only did I gain knowledge of this subject but I also gained a genuine passion for a developing technology that continues to present new and fascinating ideas. By undergoing this project I have developed a better idea of the capabilities of machine learning and how important having accurate data is to creating a functioning system.

Bibliography

Ableton, 2015. *Ableton*. [Online] Available at: https://www.ableton.com/ [Accessed 1 January 2015].

Alexander Galembo, A.A., 1999. Signal Representation and Estimation of Spectral Parameters by Inharmonic Comb Filters with Application to the Piano. *IEEE Transactions on speech and audio processing, VOL. 7, NO. 2, MARCH 1999*, Vol. 7(2), pp.197 - 203.

Boss, 2015. *GP-10 Guitar Processor*. [Online] Available at: http://www.bossus.com/products/gp-10/ [Accessed 1 January 2015].

Capo, 2015. *The future of learning to play.* [Online] Available at: http://supermegaultragroovy.com/products/capo/ [Accessed 1 January 2015].

Chih-Chung Chang, C.-J.L., 2014. *LIBSVM -- A Library for Support Vector Machines*. [Online] (3.20) Available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm/ [Accessed 1 January 2015].

Cohen, J., 2014. *Machine translation of guitar audio*. Final year project. Cardiff University.

Indiginus, 2015. *Indiginus sample libraries*. [Online] Available at: http://www.indiginus.com/agc.html [Accessed 1 January 2015].

Ng, A., 2014. *Machine Learning*. [Online] Available at: https://www.coursera.org/course/ml [Accessed 1 January 2015].

Pritchard, M., 2002. *Musical analysis of audio. Master's thesis*. Cardiff University.

Reisinger, D., 2011. *Guitar Hero tops list of best-selling games*. [Online] Available at: http://www.cnet.com/uk/news/guitar-hero-tops-list-of-best-selling-games/ [Accessed 1 January 2015].

Schutte, K., 2012. *Matlab and MIDI*. [Online] Available at: http://www.kenschutte.com/midi [Accessed 1 January 2015].

Toiviainen, P., 2004. *Midi Toolbox*. [Online] Available at: https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/miditoolbox/ [Accessed 1 January 2015].

Watson, K., 2014. *IMS Business report*. Ibizs International Music Summit.

Table of Figures

Appendix

A: Multi Guitar Type Confusion Matrices / ROC Curves

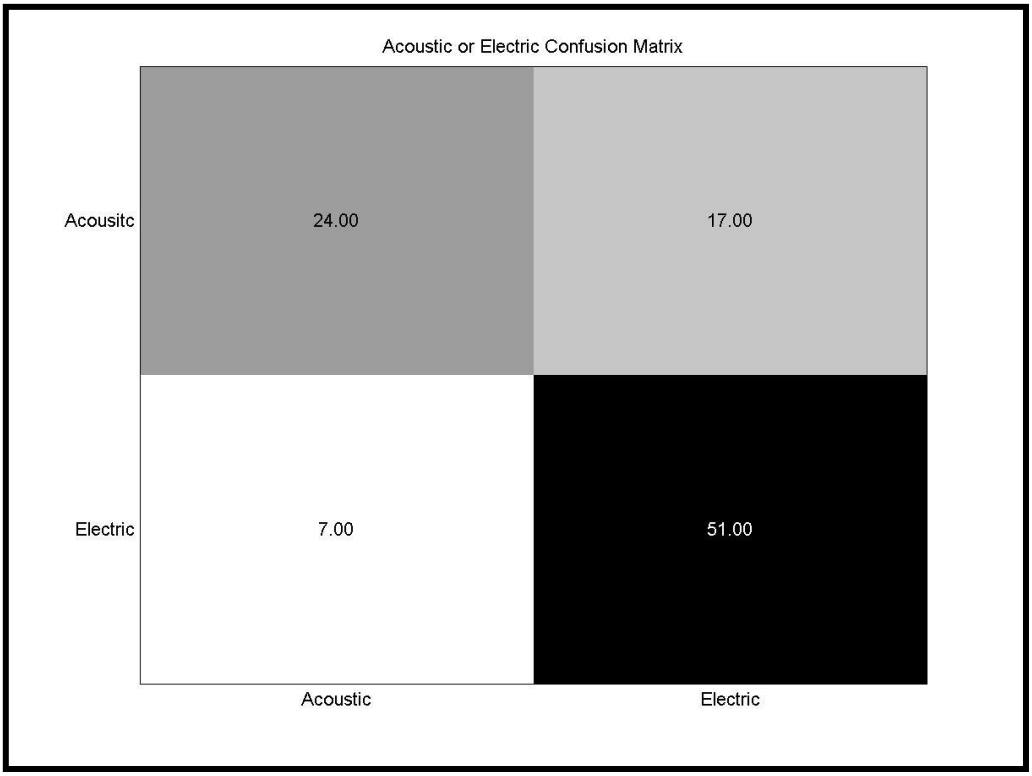

Figure 10 - Acoustic Or Electric Confusion Matrix

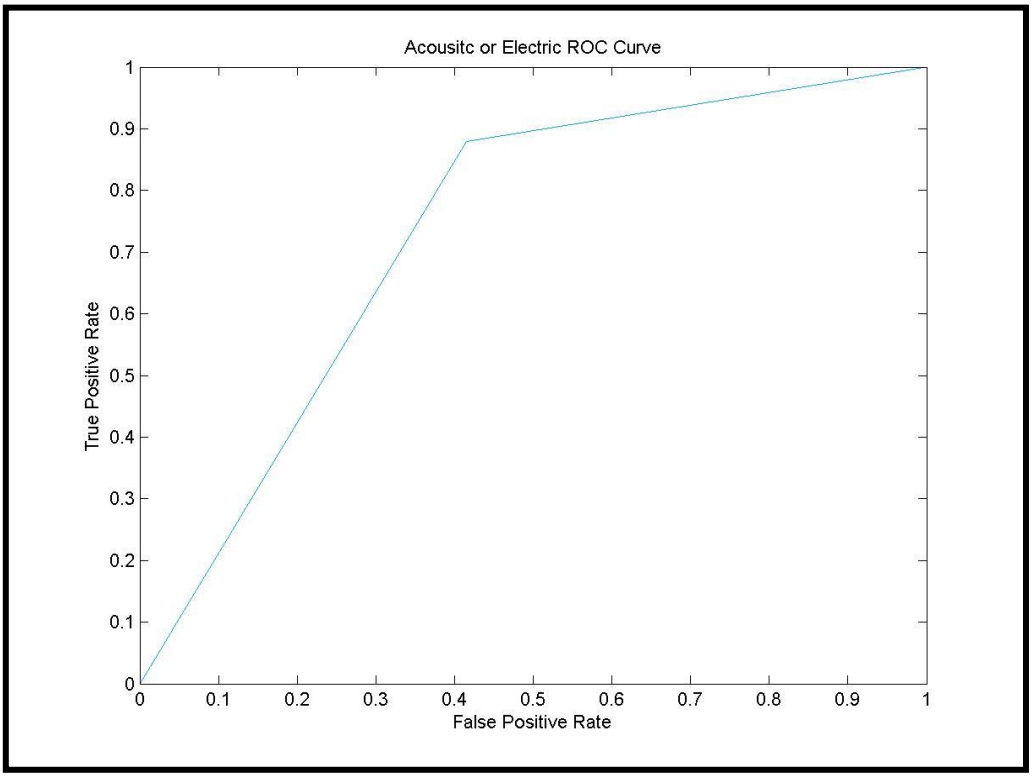

Figure 11 - Acoustic or Electric ROC Curve

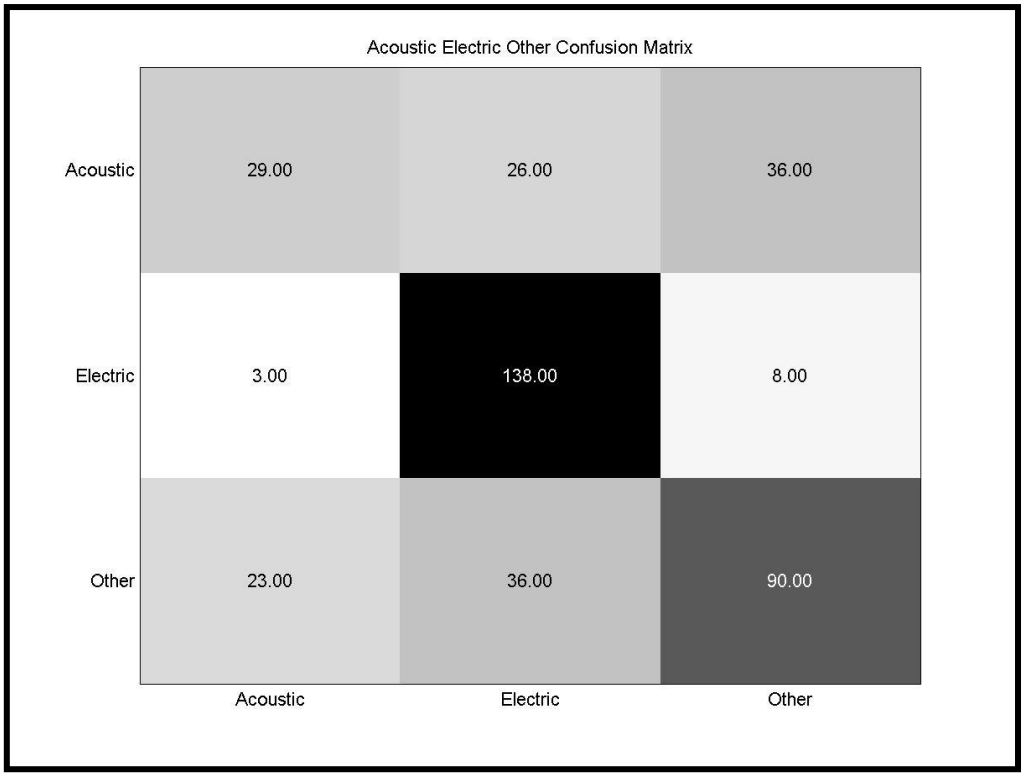

Figure 12 - Guitar Type Acoustic Electric or Other Confusion Matrix

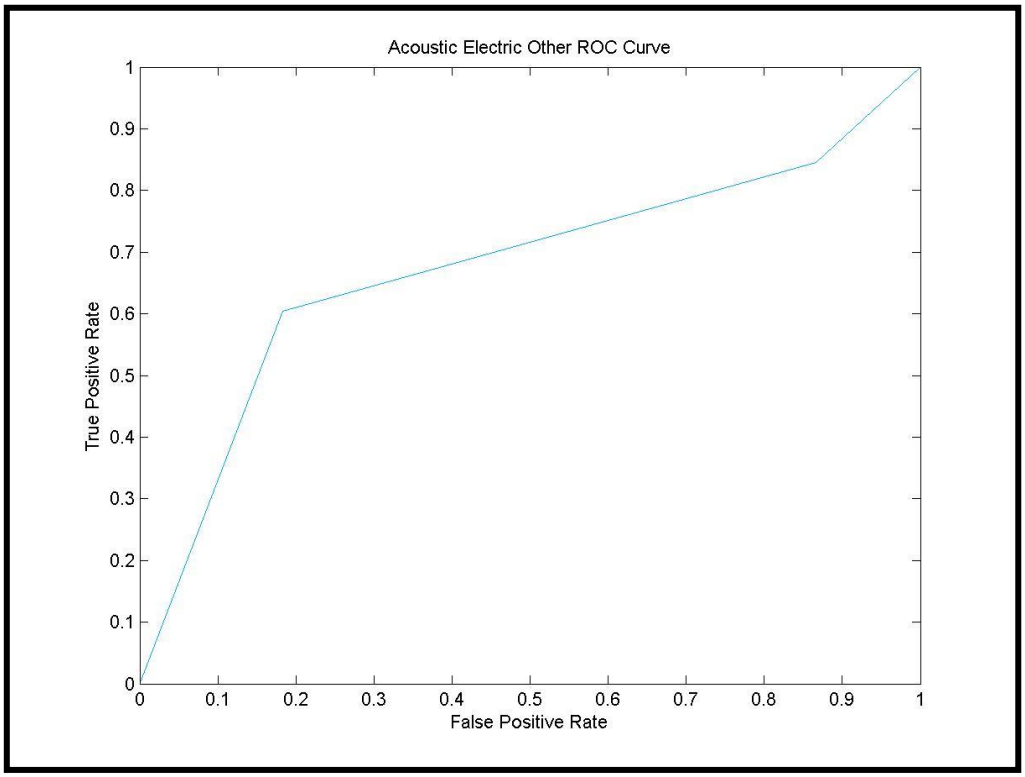

Figure 13 - Guitar Type Acoustic Electric or Other ROC Curve

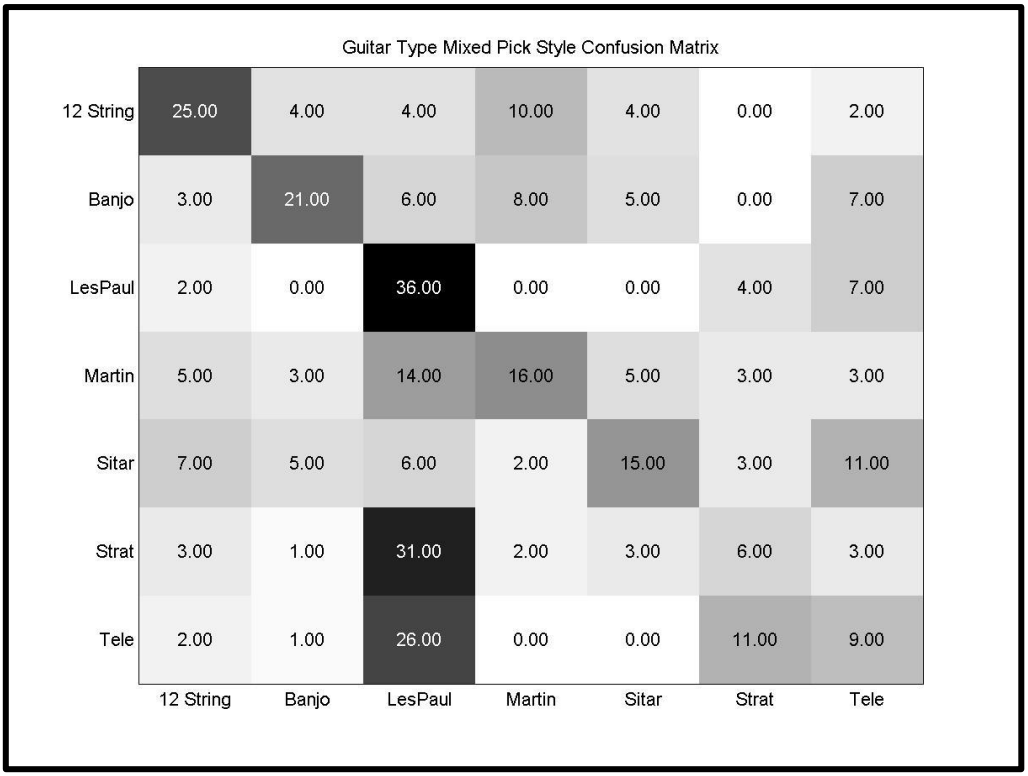

Figure 14 - Guitar Type Mixed Pick Style Confusion Matrix

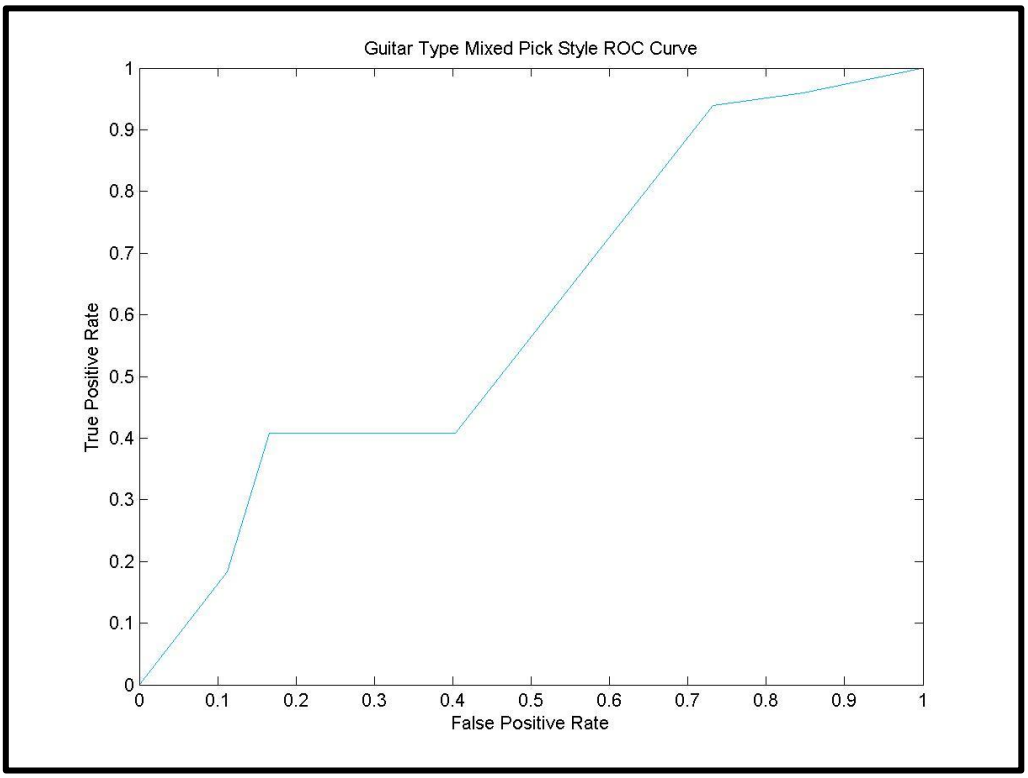

Figure 15 - Guitar Type Mixed Pick Style ROC Curve

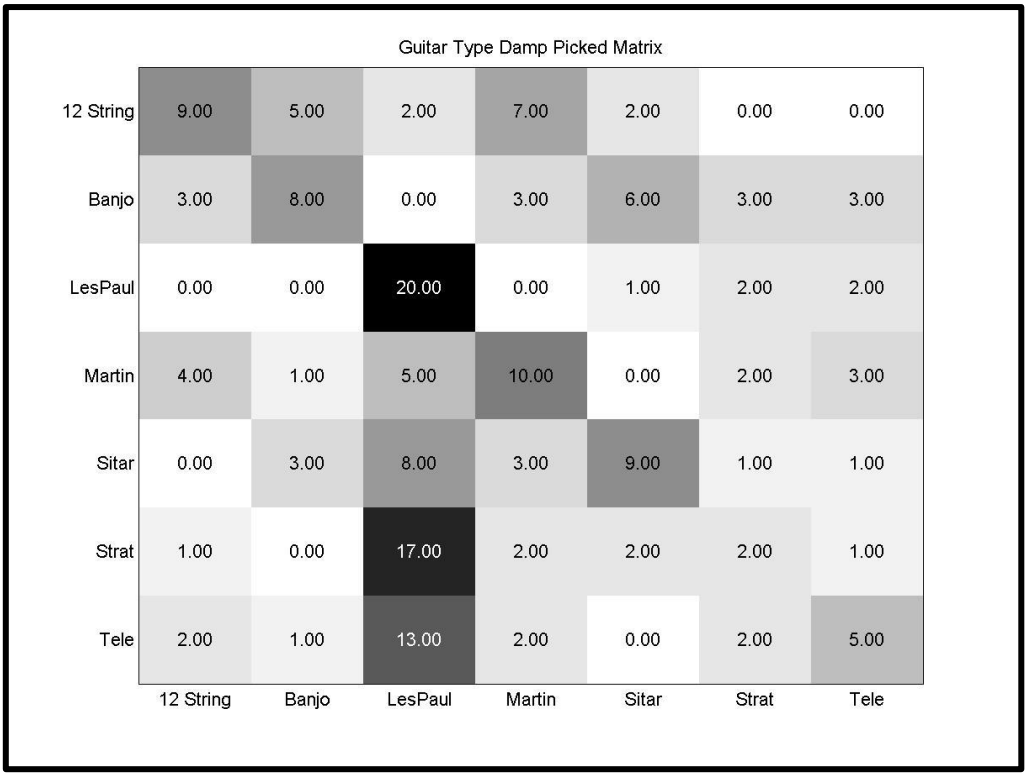

Figure 16 - Guitar Type Damp Picked Confusion Matrix

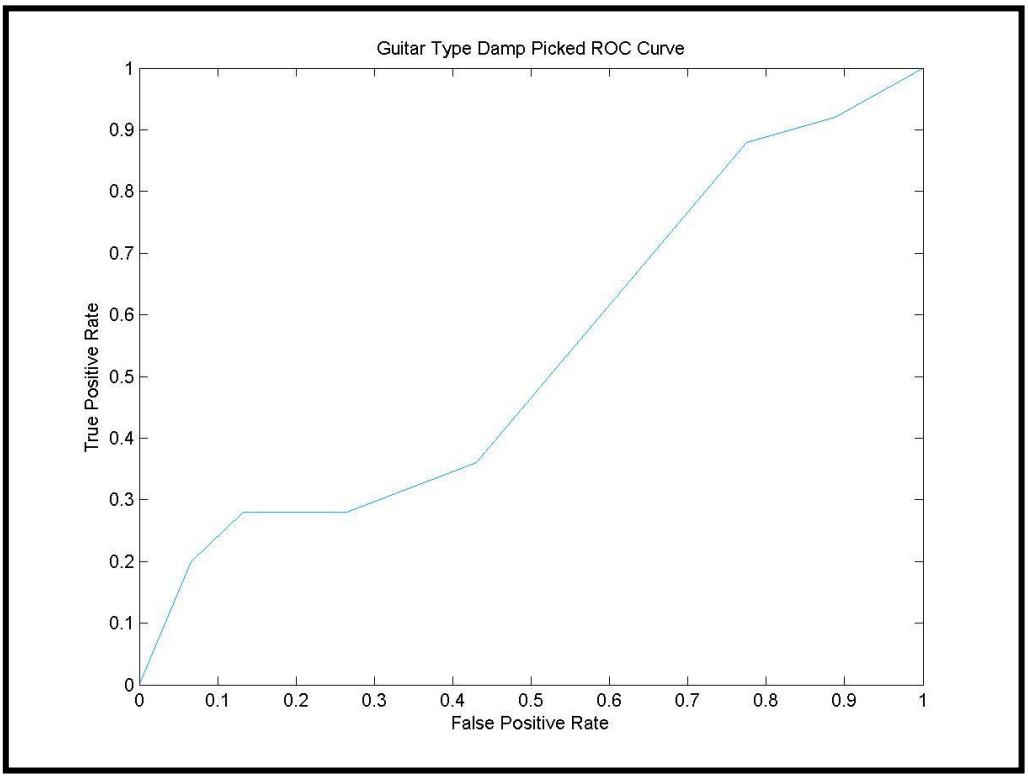

Figure 17 - Guitar Type Damp Picked ROC Curve

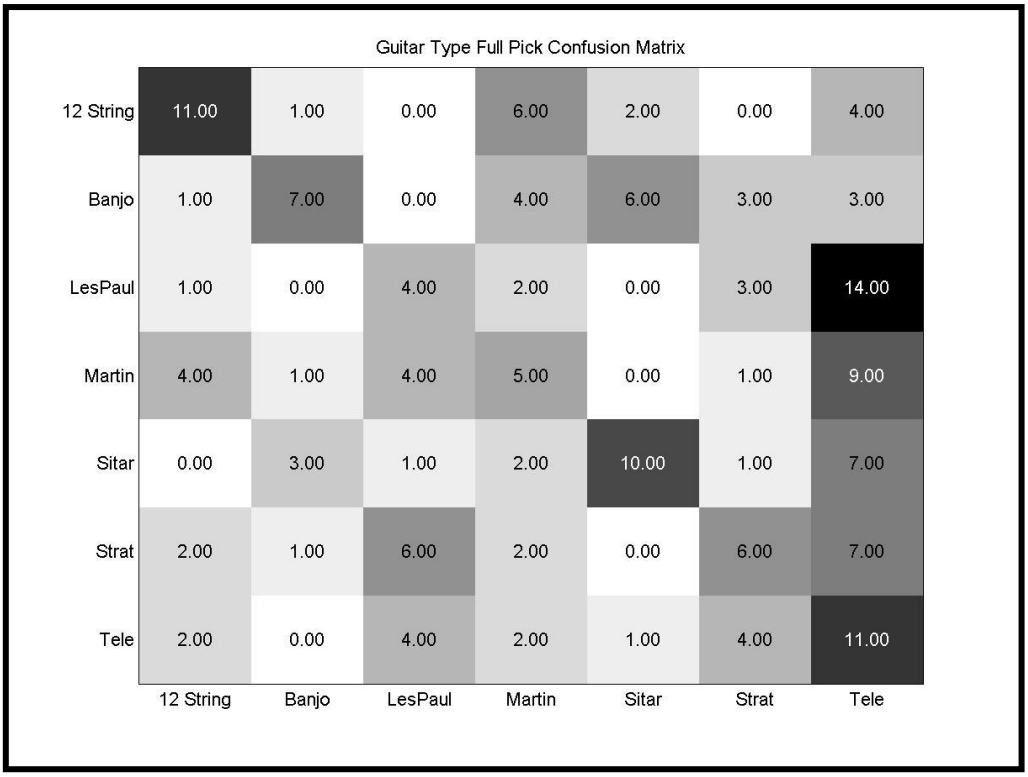

Figure 18 - Guitar Type Full Pick Confusion Matrix

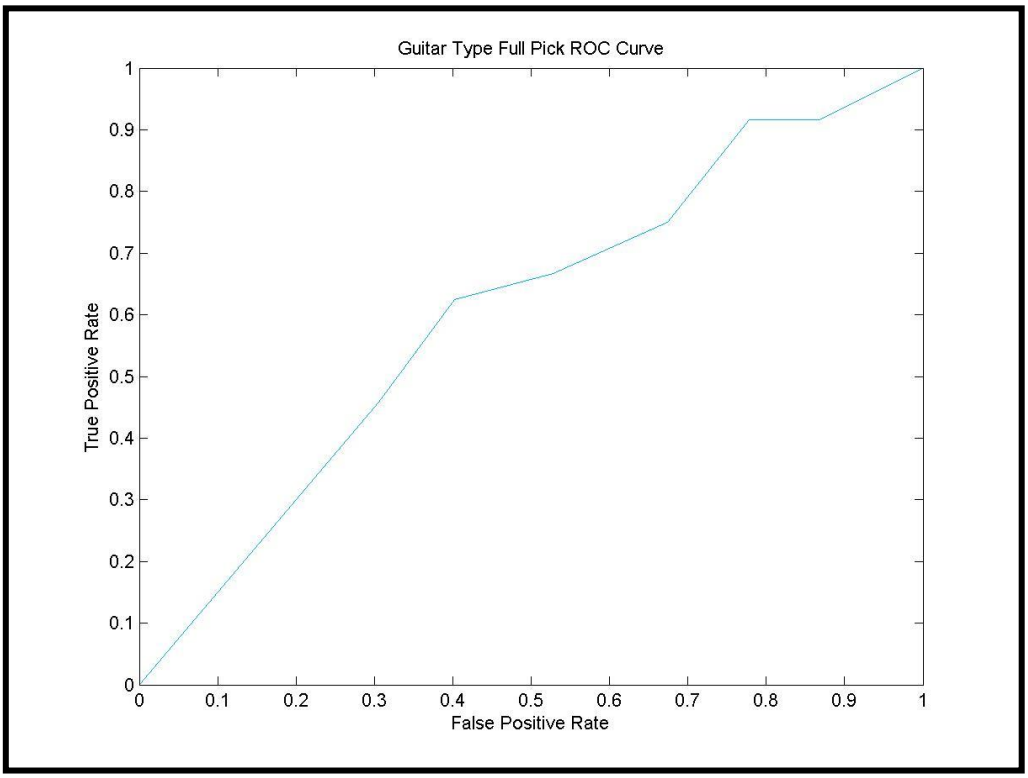

Figure 19 - Guitar Type Full Pick ROC Curve

B: Single Guitar Type Confusion Matrices / ROC Curves

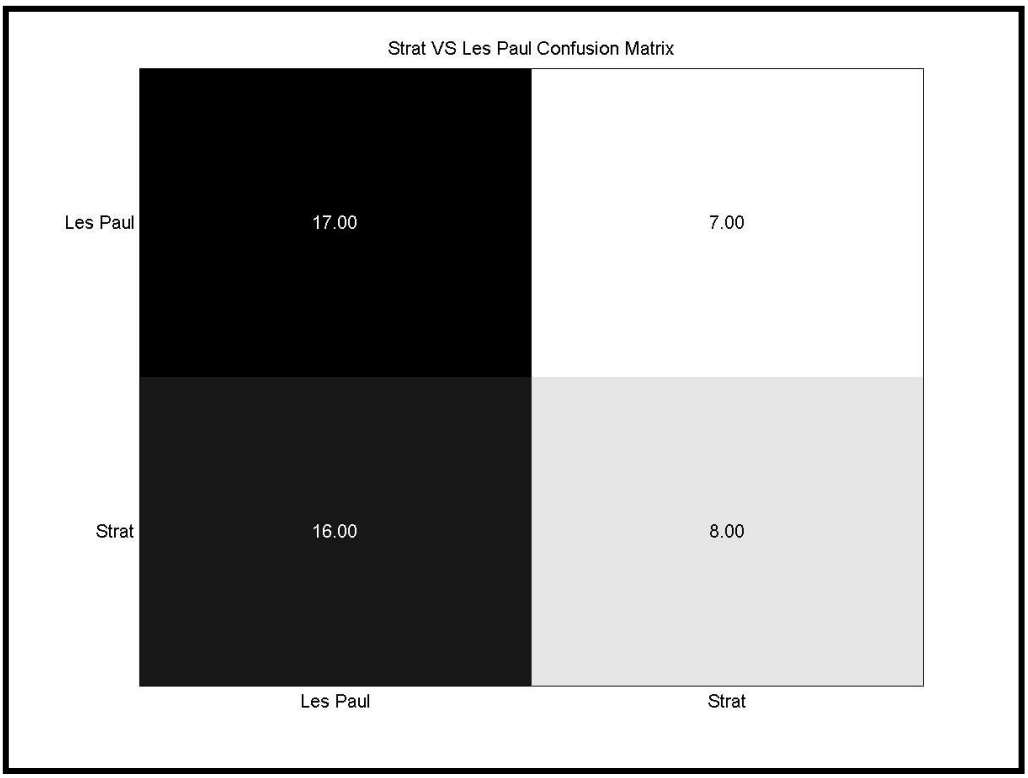

Figure 20 - Strat Vs Les Paul Confusion Matrix

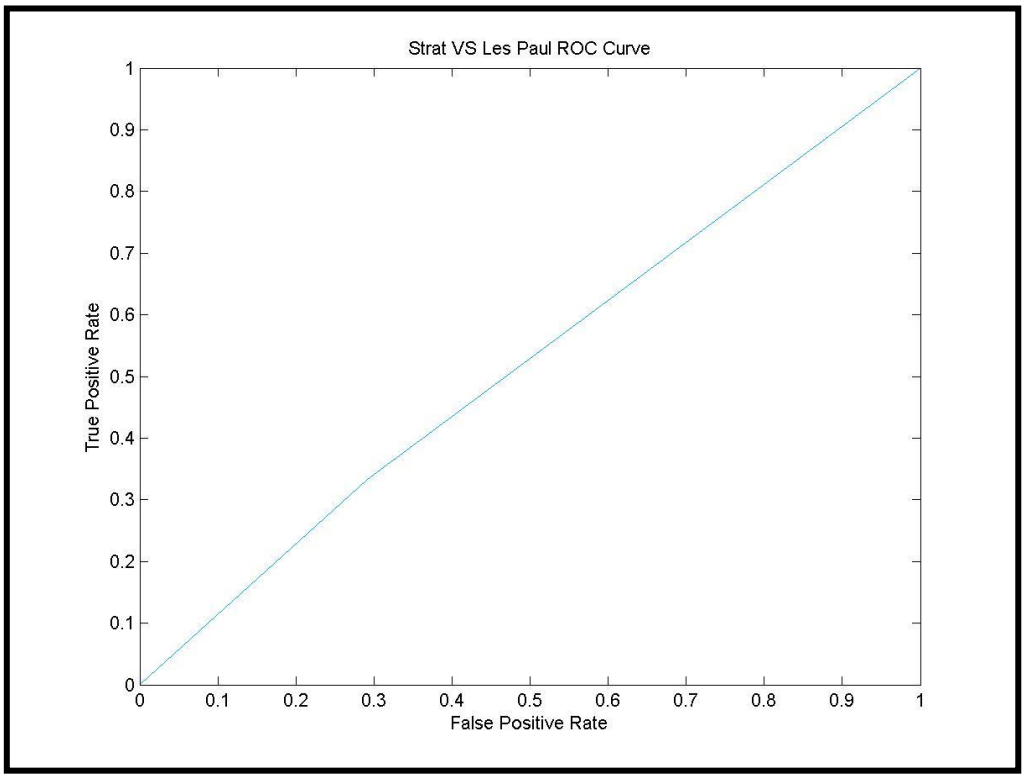

Figure 21 - Strat Vs Les Paul ROC Curve

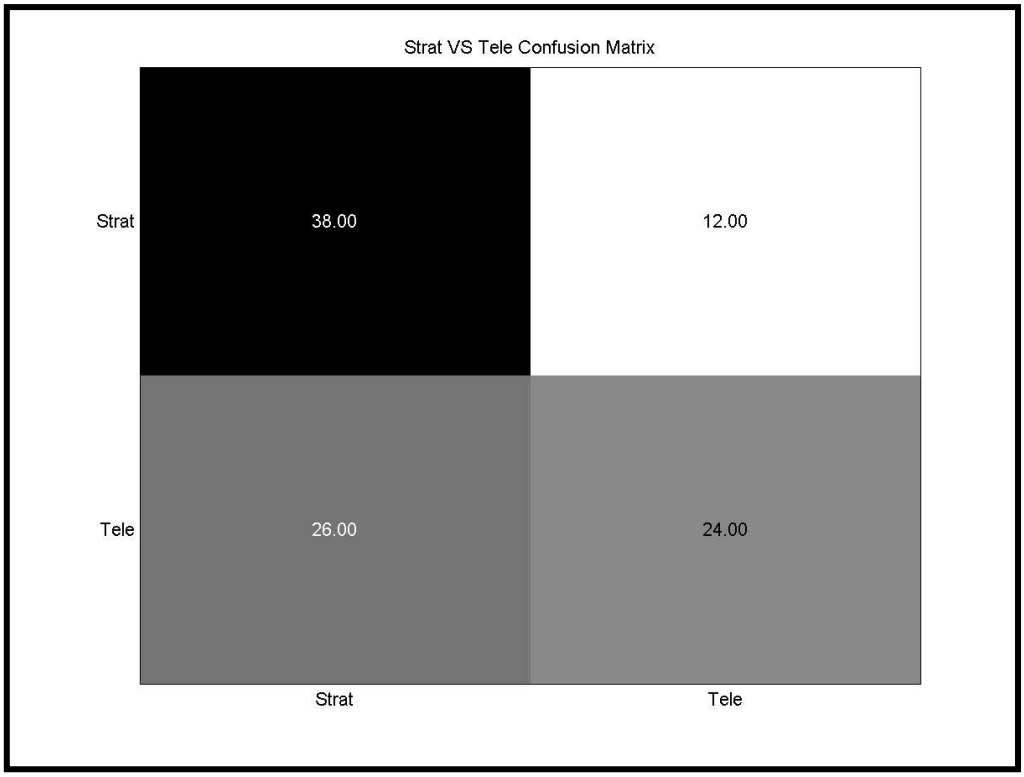

Figure 22 - Strat Vs Tele Confusion Matrix

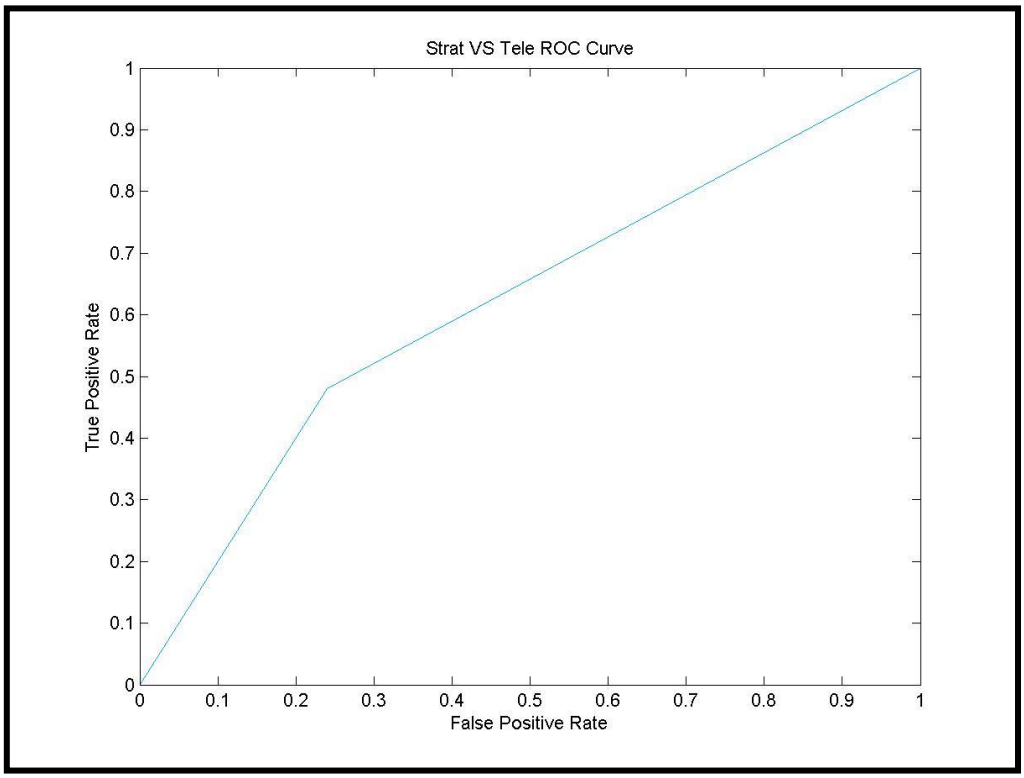

Figure 23 - Strat Vs Tele ROC Curve

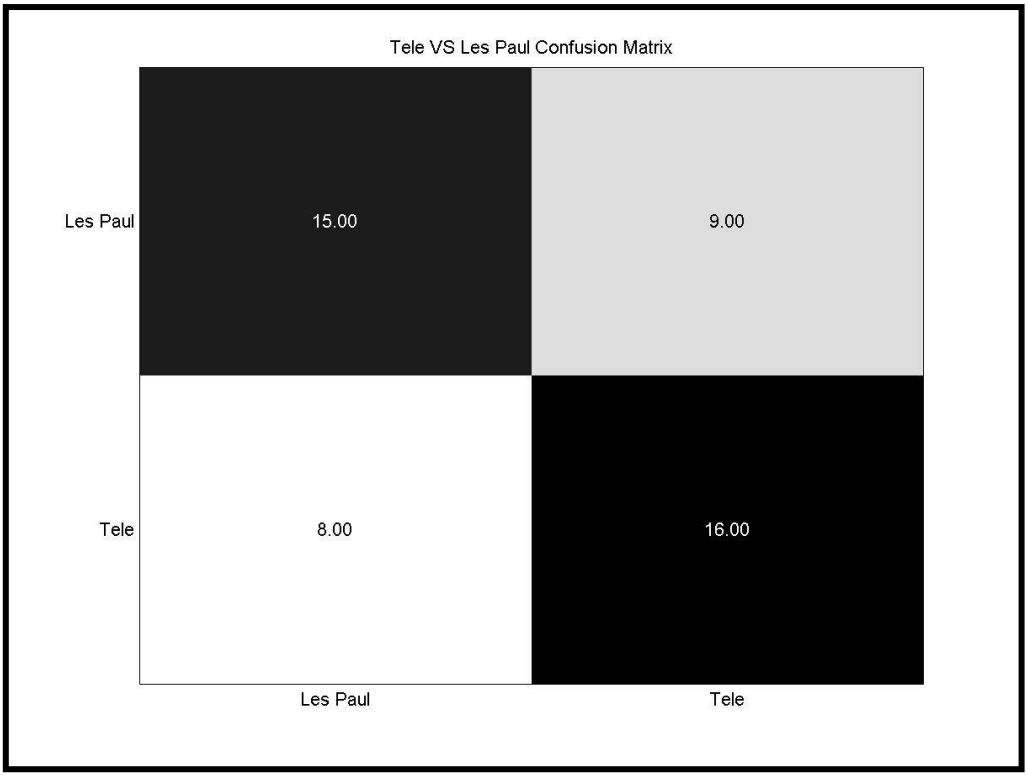

Figure 24 - Tele Vs Les Paul Confusion Matrix

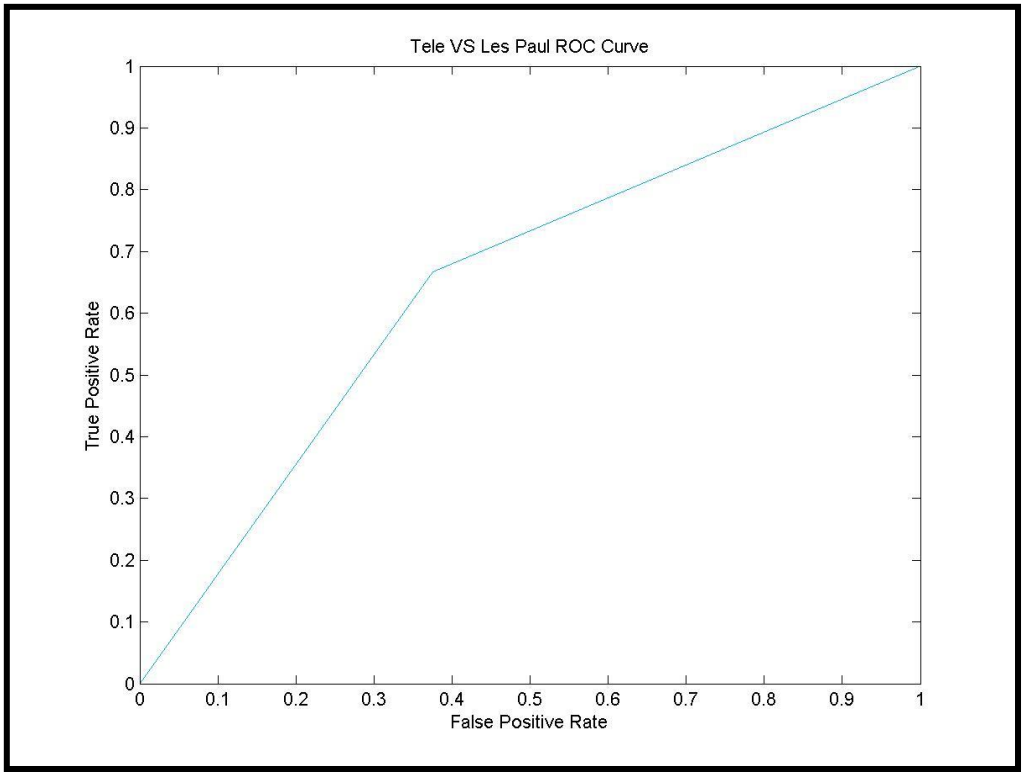

Figure 25 - Tele Vs Les Paul ROC Curve

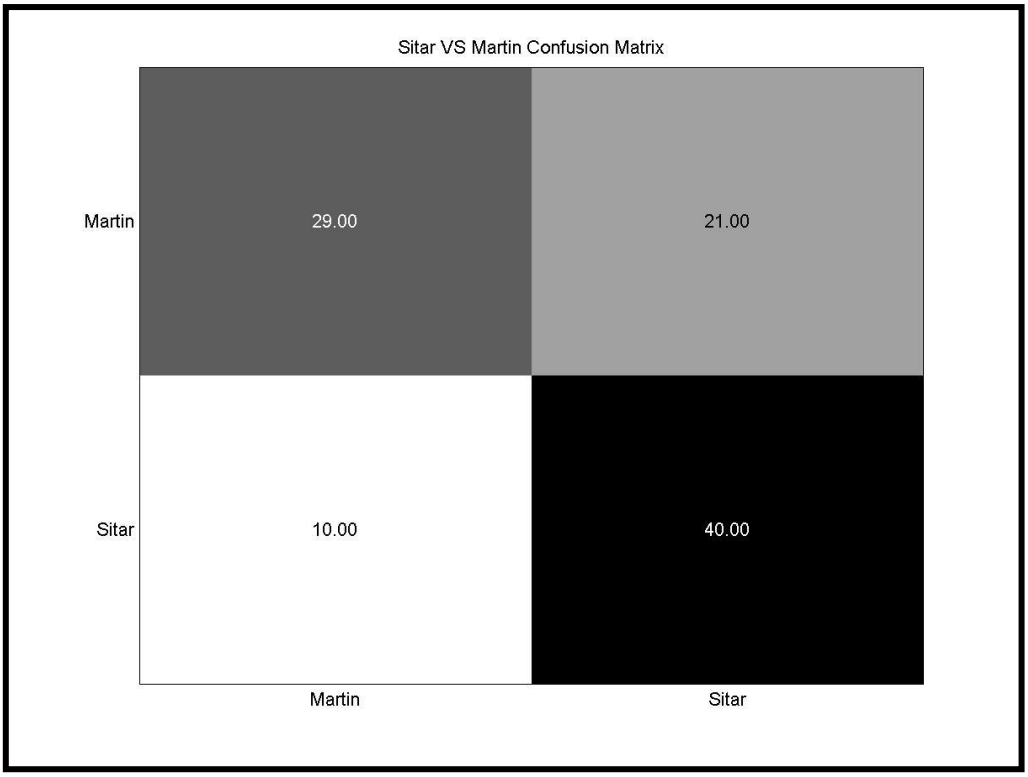

Figure 26 - Sitar Vs Martin Confusion Matrix

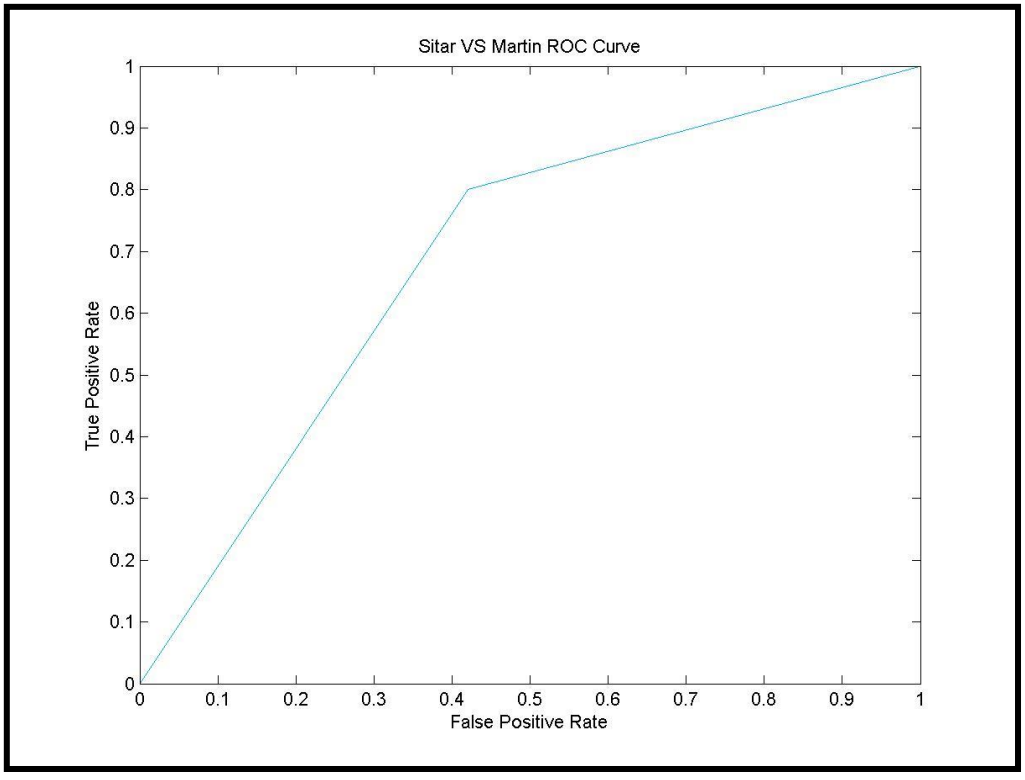

Figure 27 - Sitar Vs Martin ROC Curve

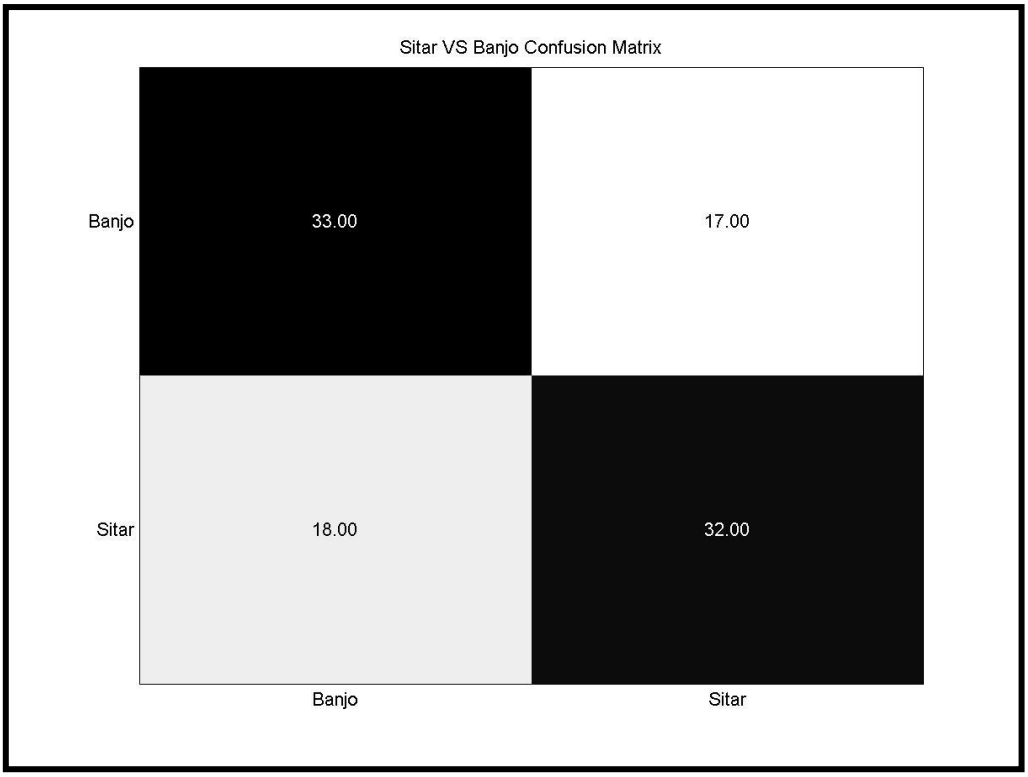

Figure 28 - SItar Vs Banjo Confusion Matrix

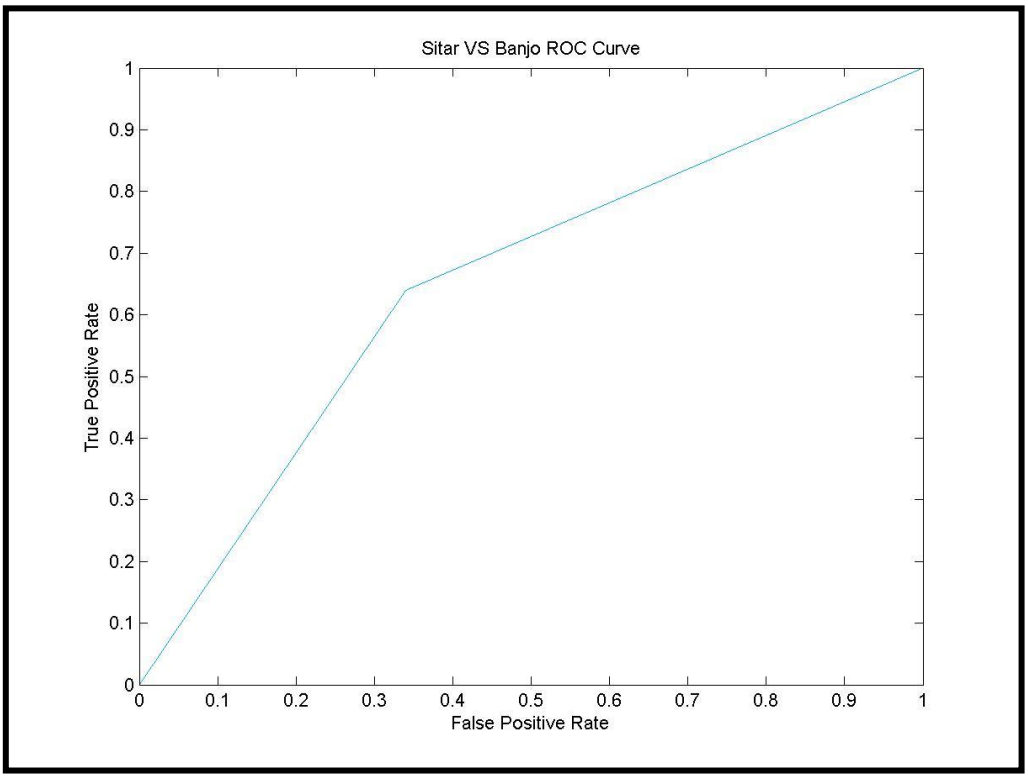

Figure 29 - Sitar Vs Banjo ROC Curve

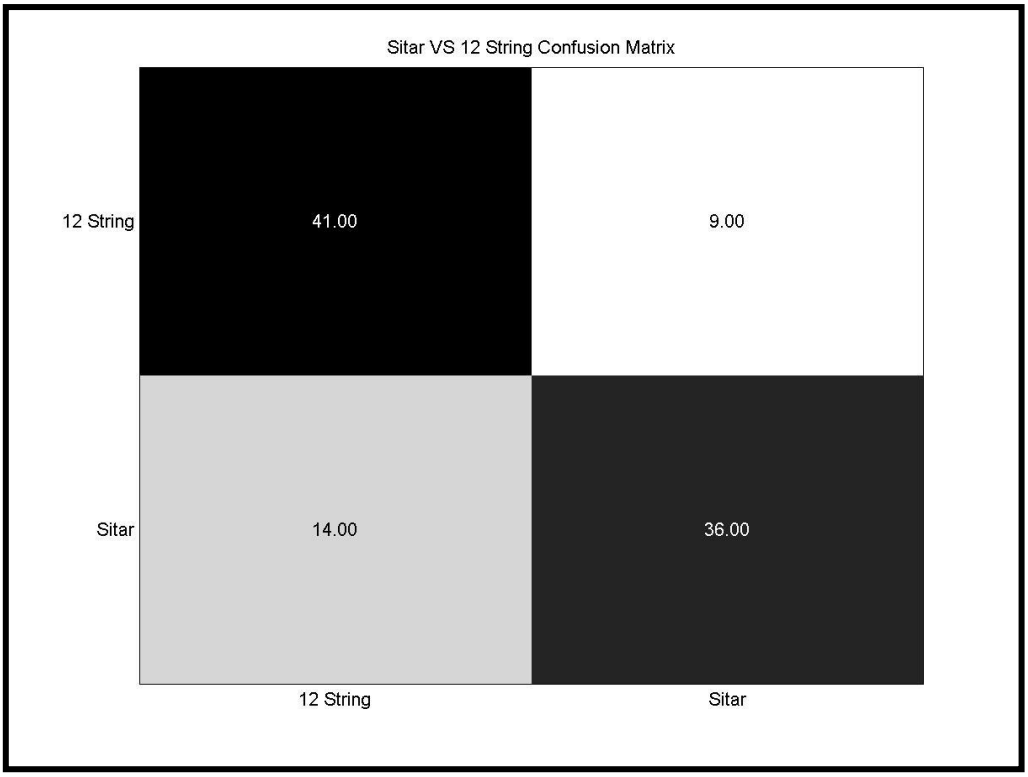

Figure 30 - SItar Vs 12 String Confusion Matrix

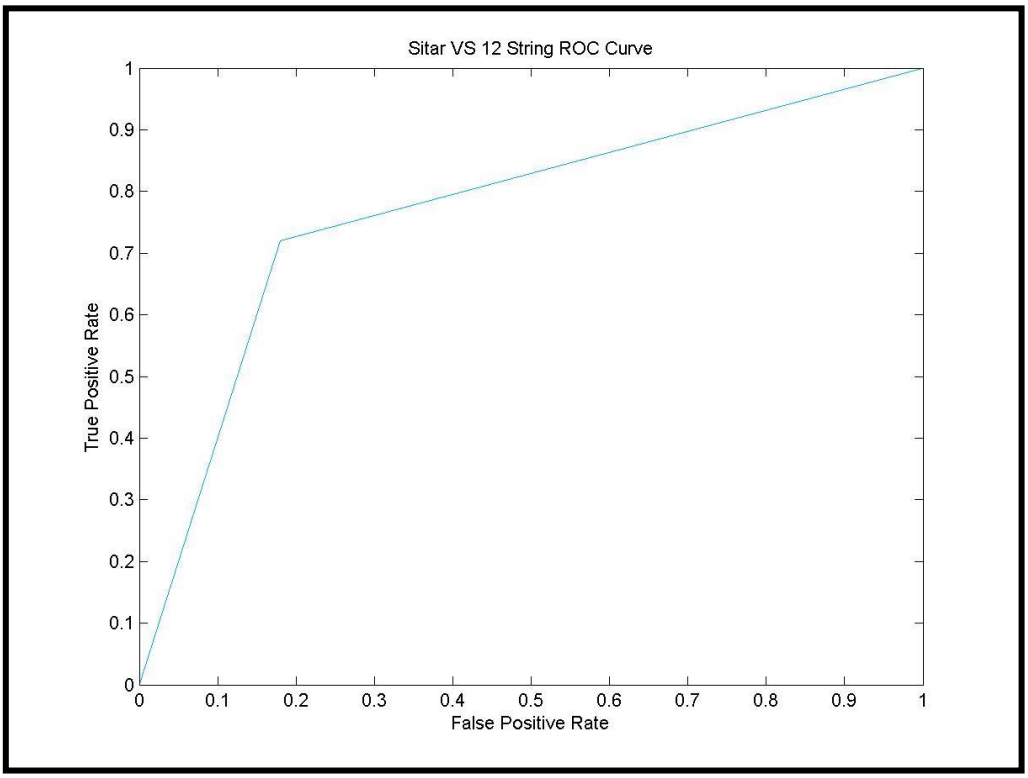

Figure 31 - Sitar Vs 12 String ROC Curve

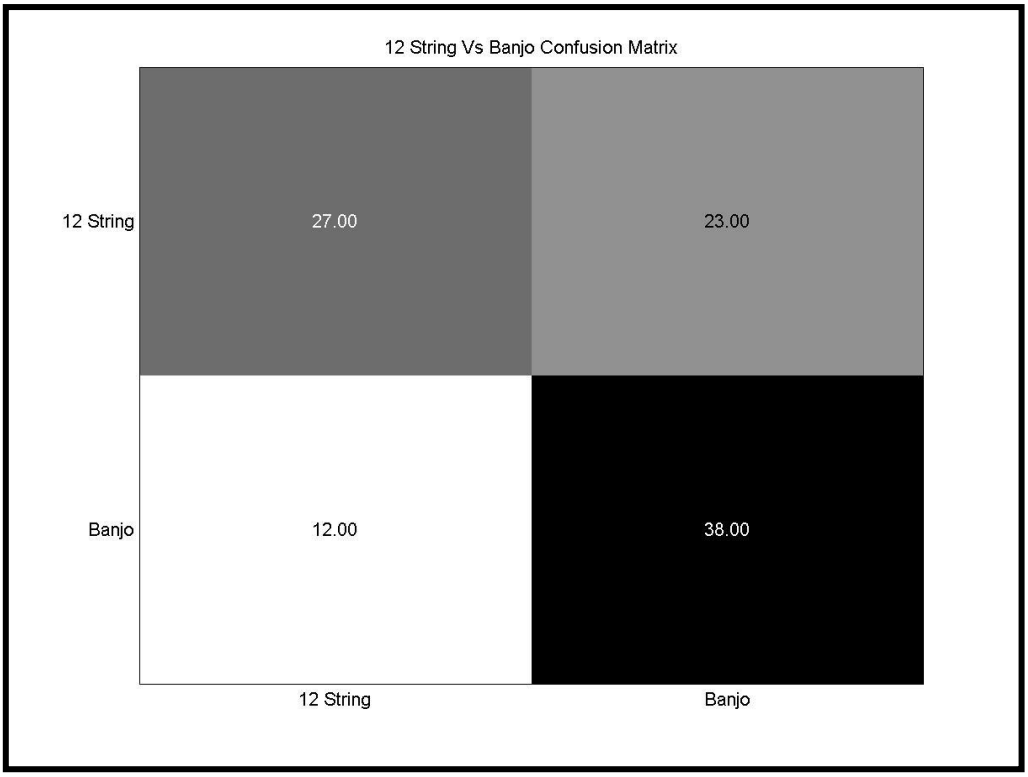

Figure 32 - 12 String Vs Banjo Confusion Matrix

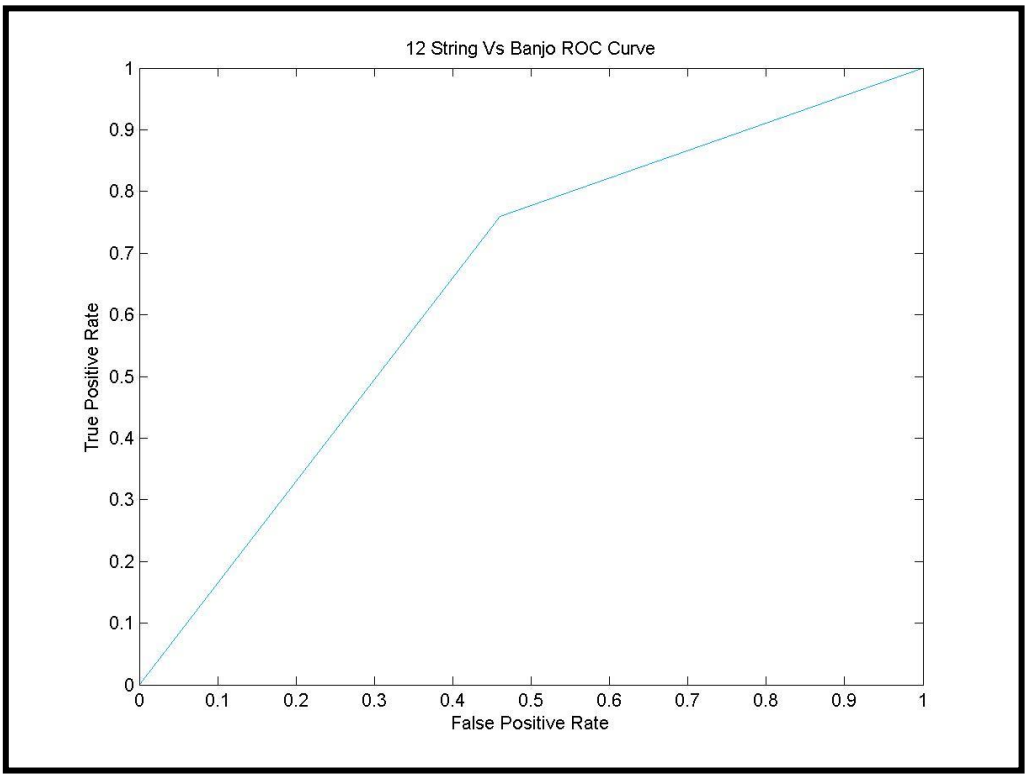

Figure 33 - 12 String Vs Banjo ROC Curve

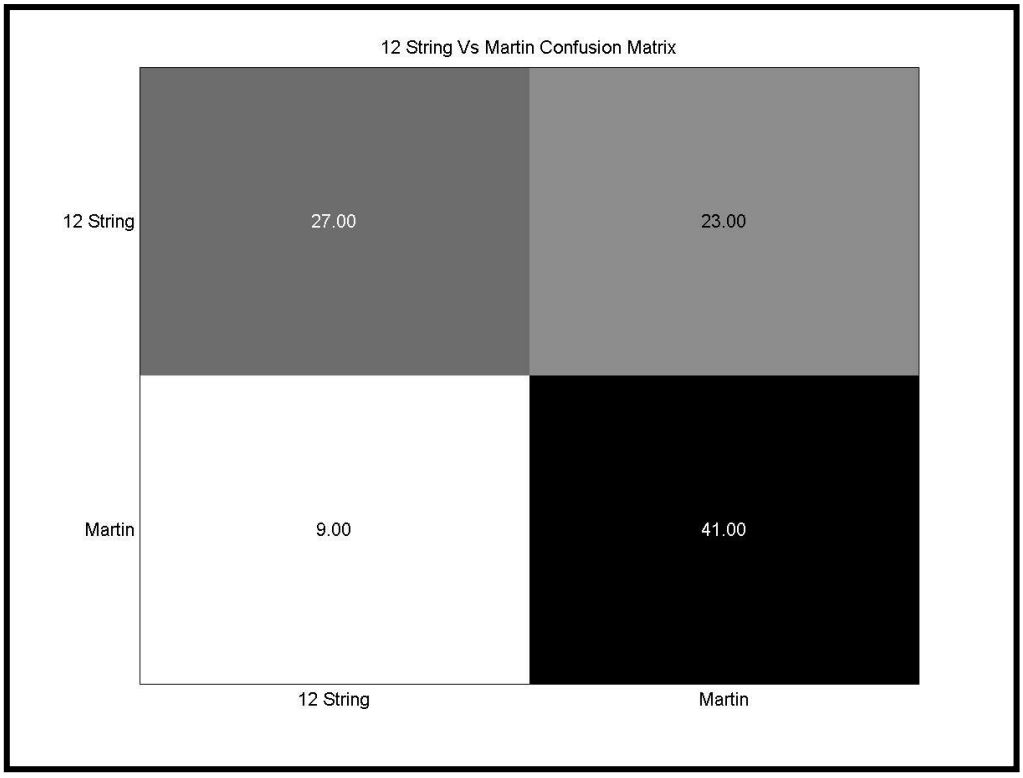

Figure 34 - 12 String Vs Martin Confusion Matrix

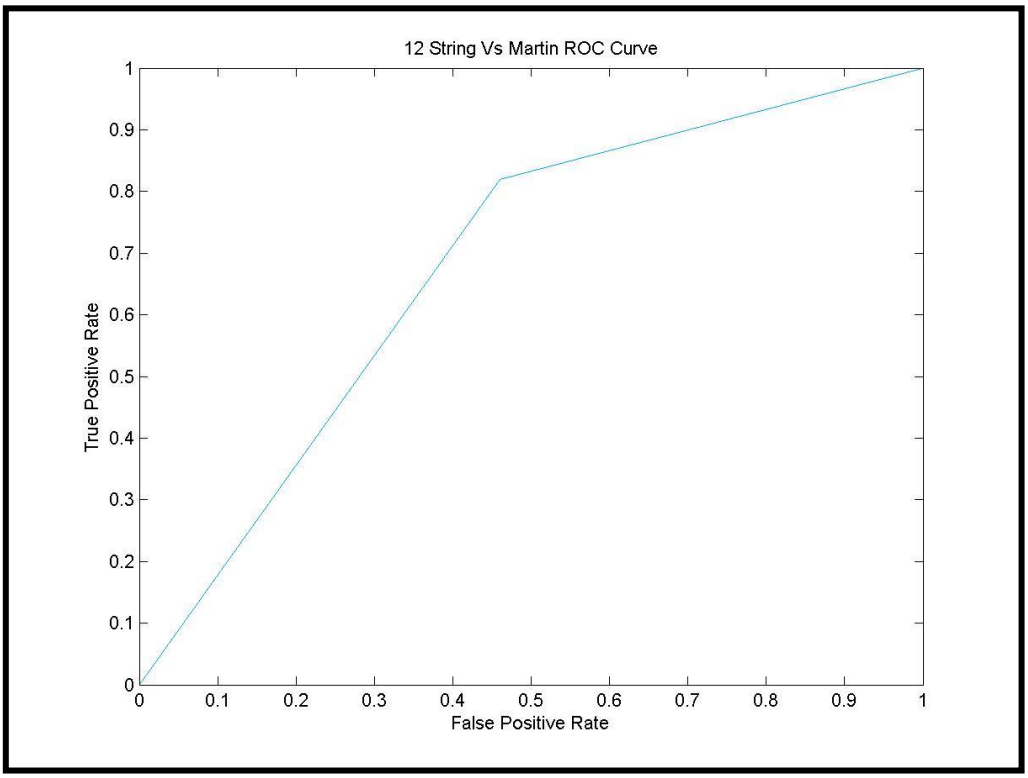

Figure 35 - 12 String Vs Martin Confusion Matrix

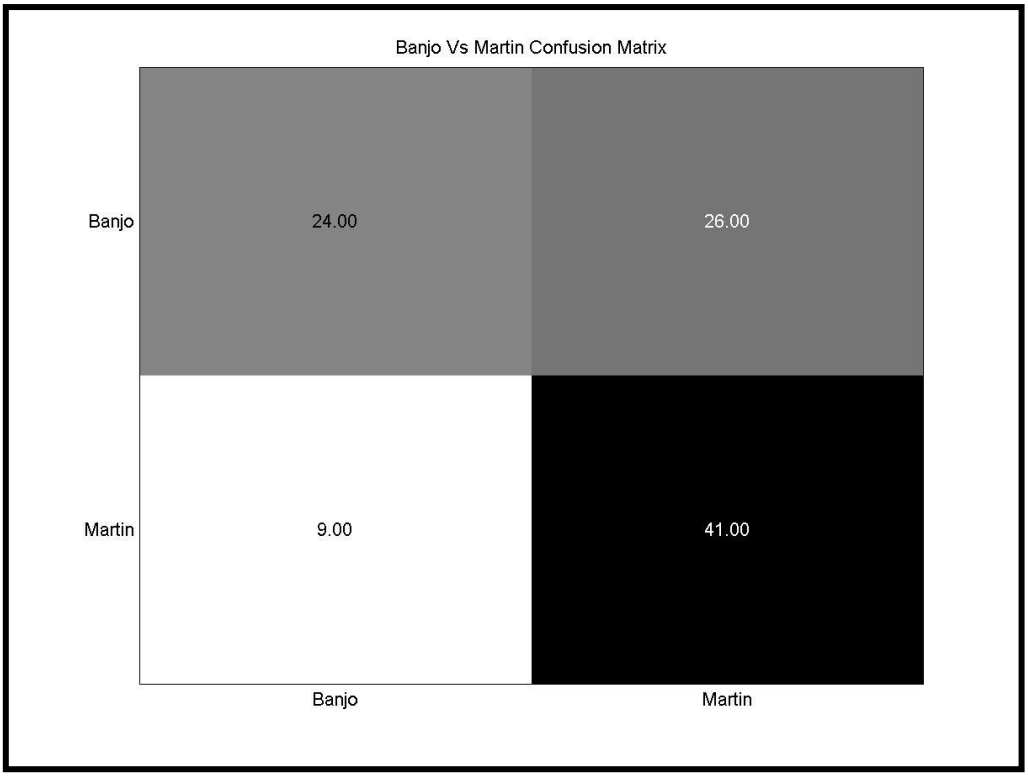

Figure 36 - Banjo Vs Martin Confusion Matrix

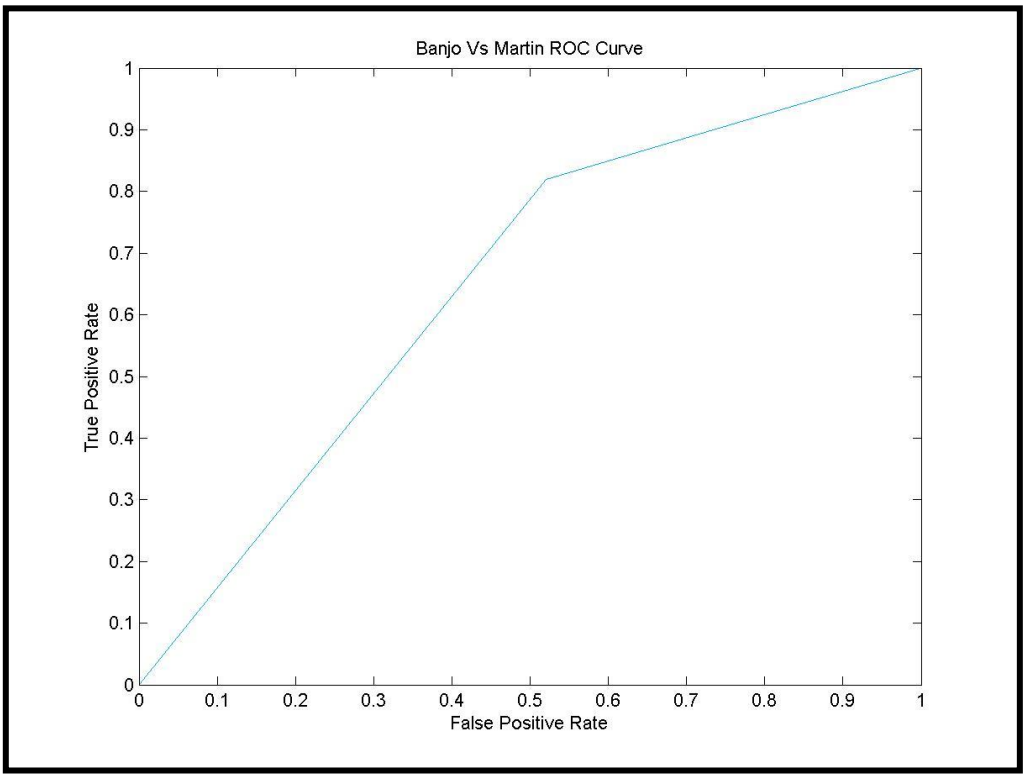

Figure 37 - Banjo Vs Martin ROC Curve

C: Vibrato Vs Non-Vibrato Confusion Matrices / ROC Curves

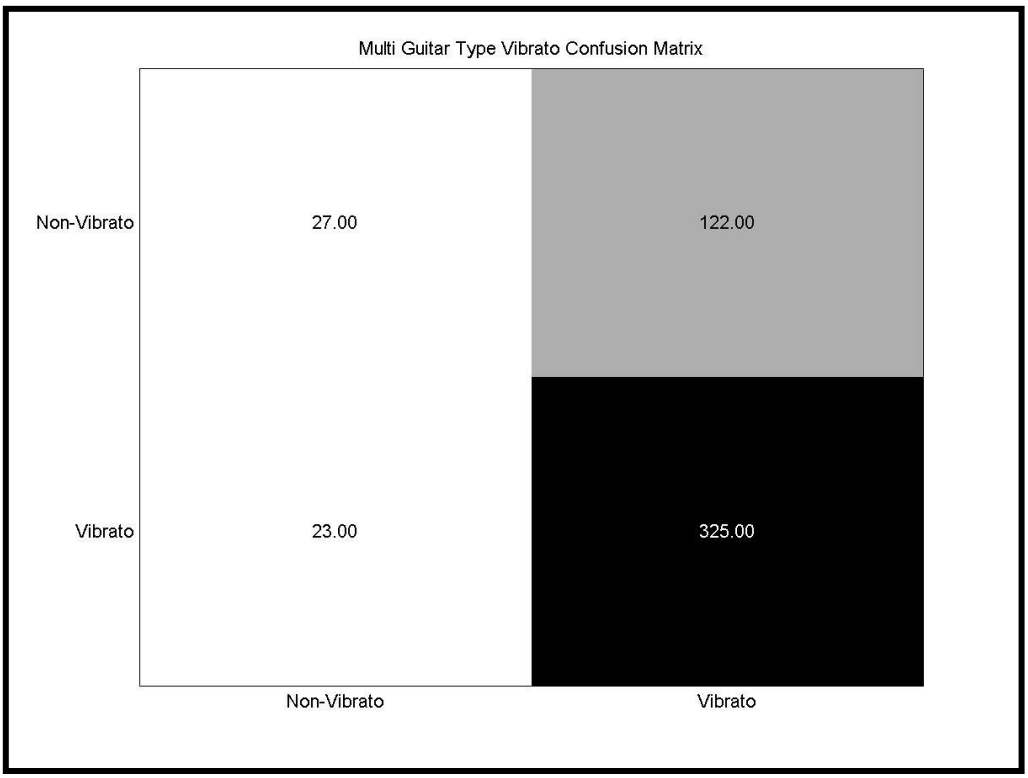

Figure 38 - Collected Guitar Type Vibrato Vs Non-Vibrato

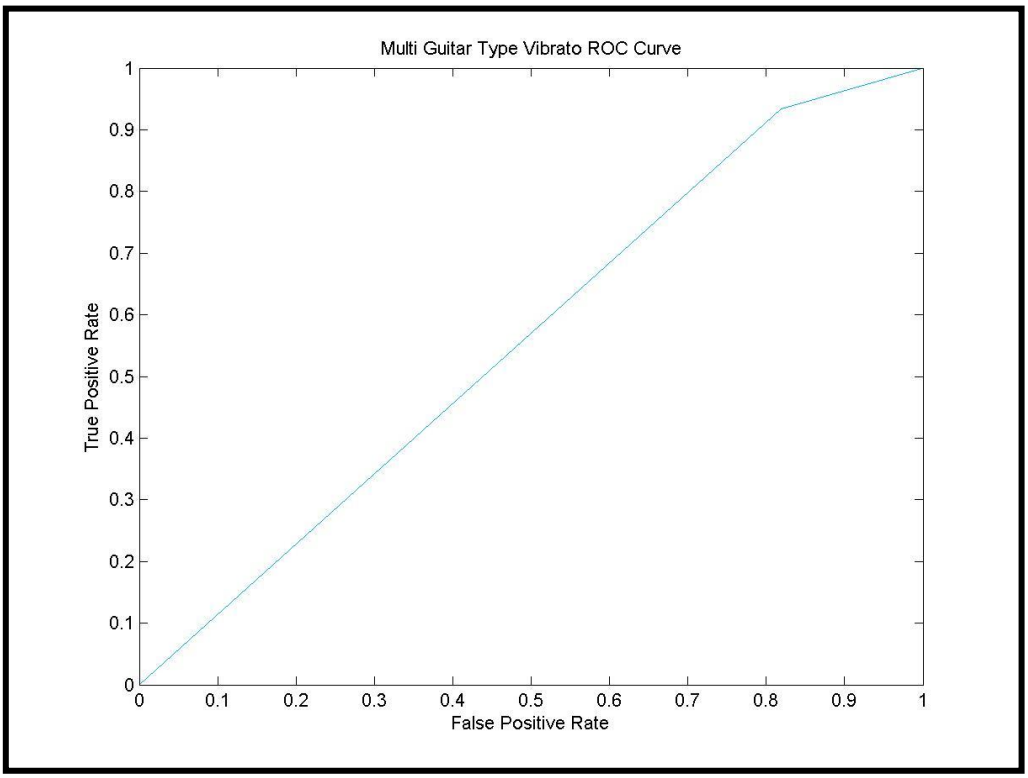

Figure 39 - Collected Guitar Type Vibrato Vs Non-Vibrato

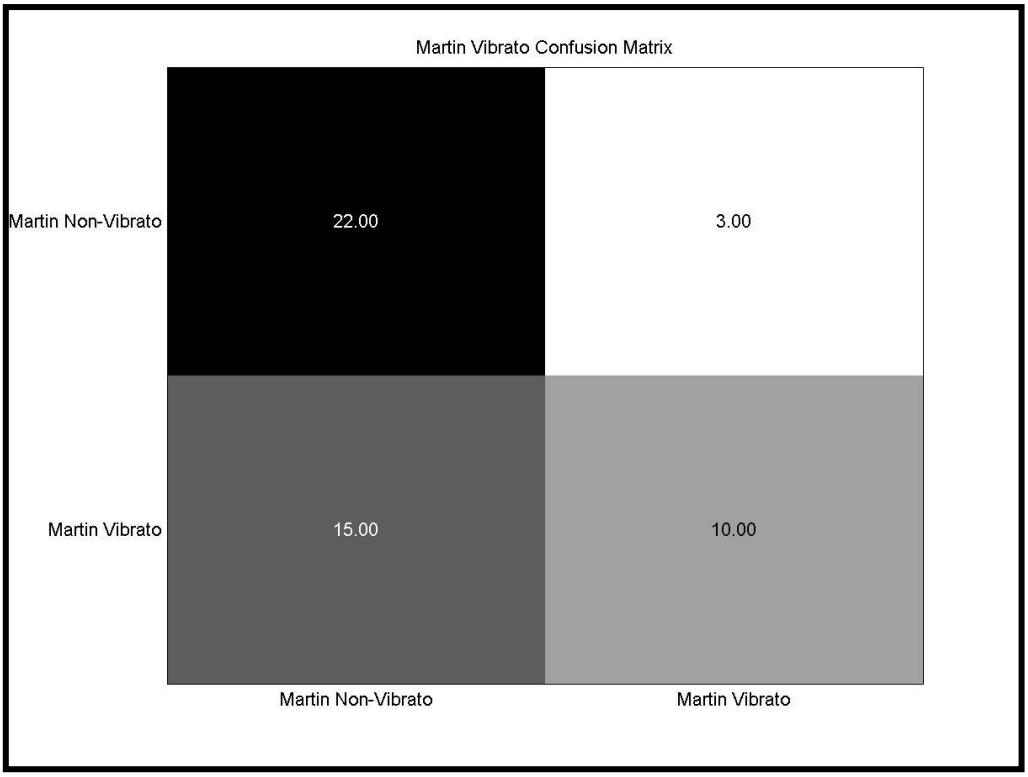

Figure 40 - Martin Vibrato Confusion Matrix

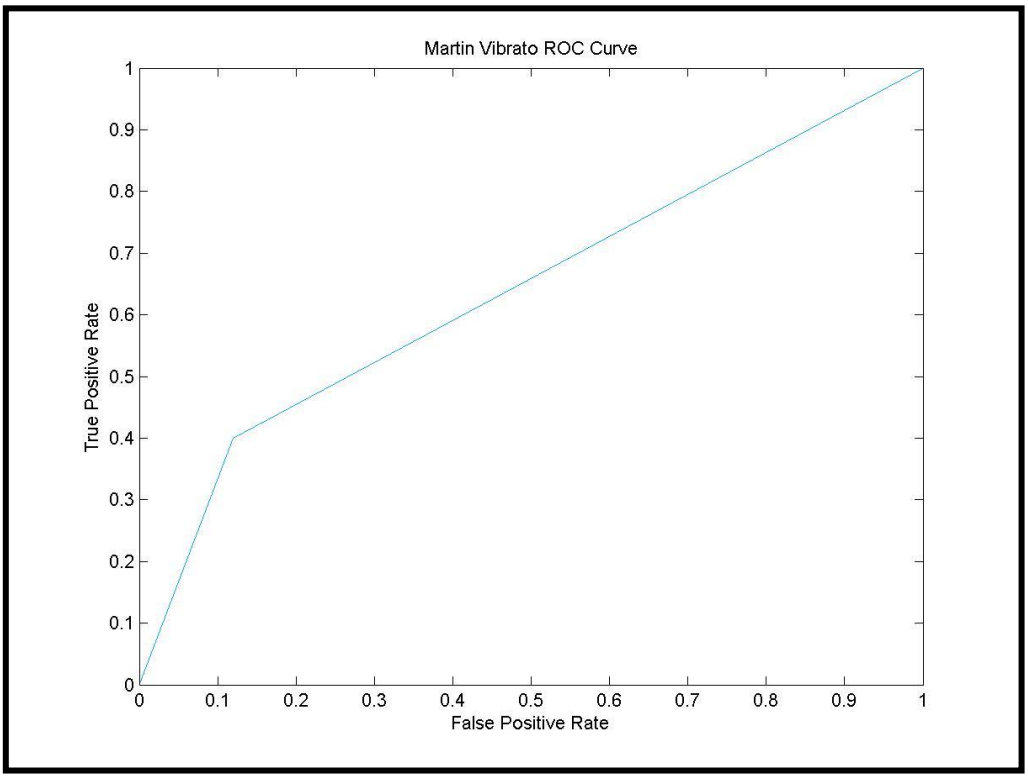

Figure 41 - Martin Vibrato ROC Curve

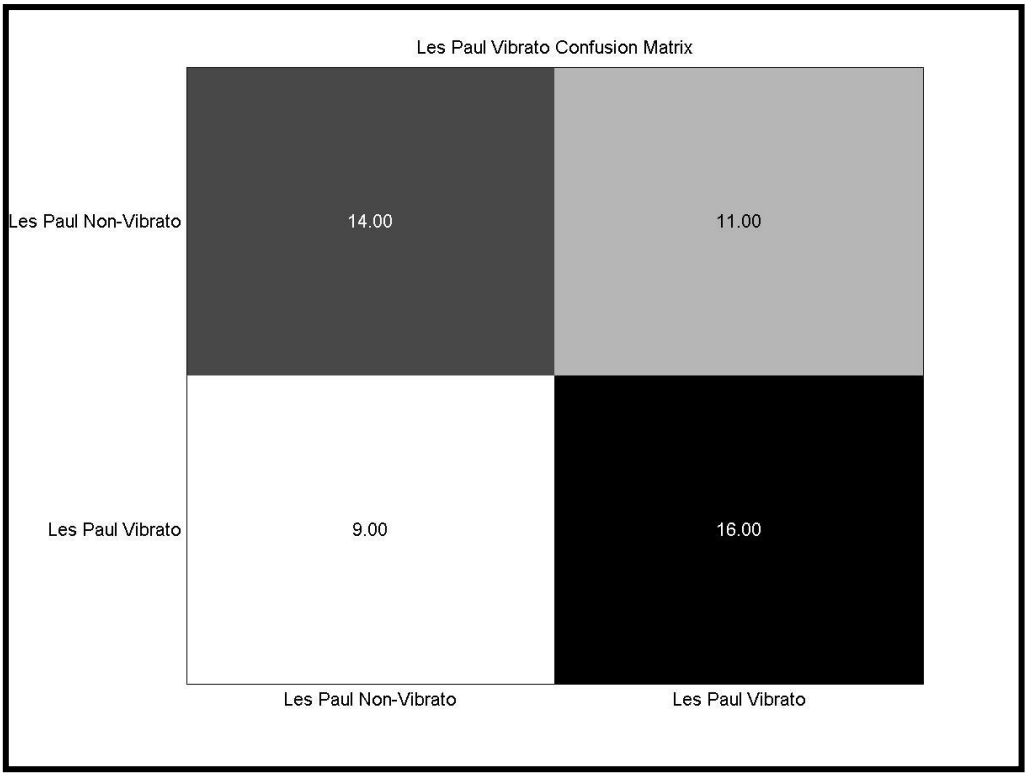

Figure 42 - Les Paul Vibrato Confusion Matrix

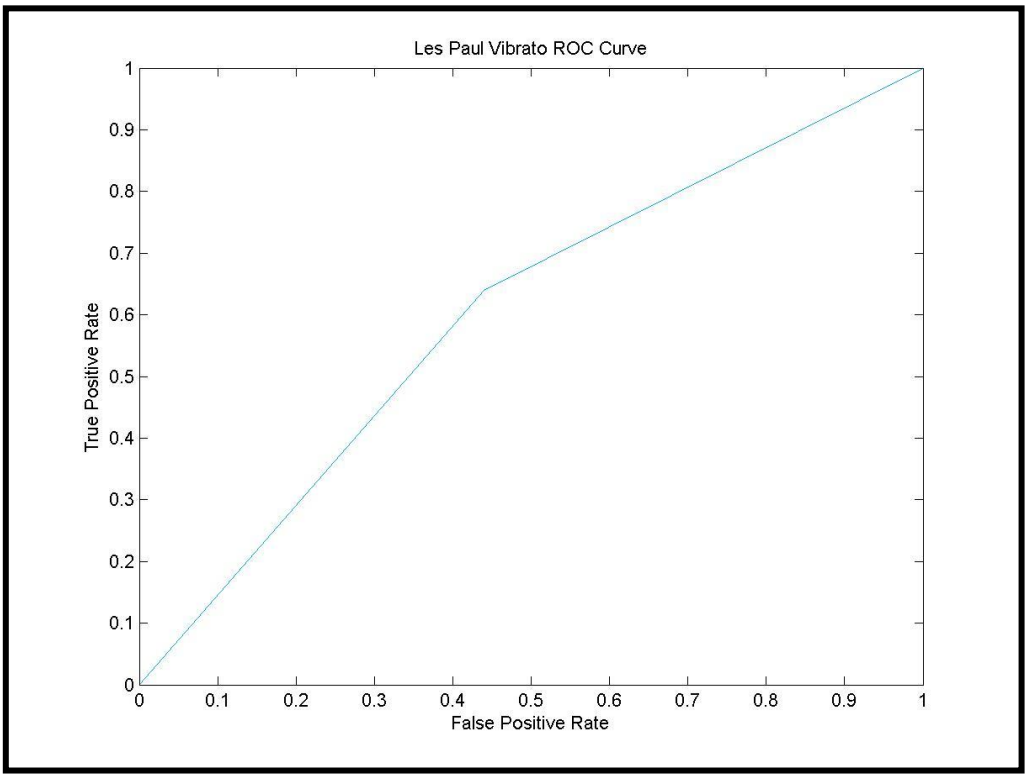

Figure 43 - Les Paul Vibrato ROC Curve

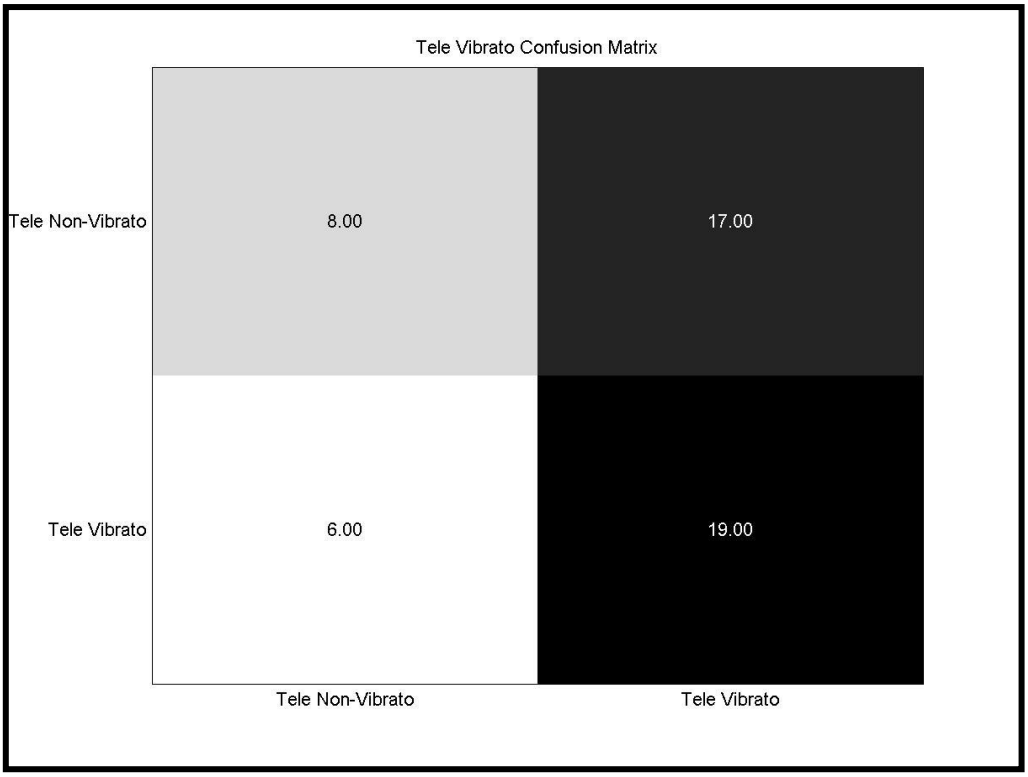

Figure 44 - Tele Vibrato Confusion Matrix

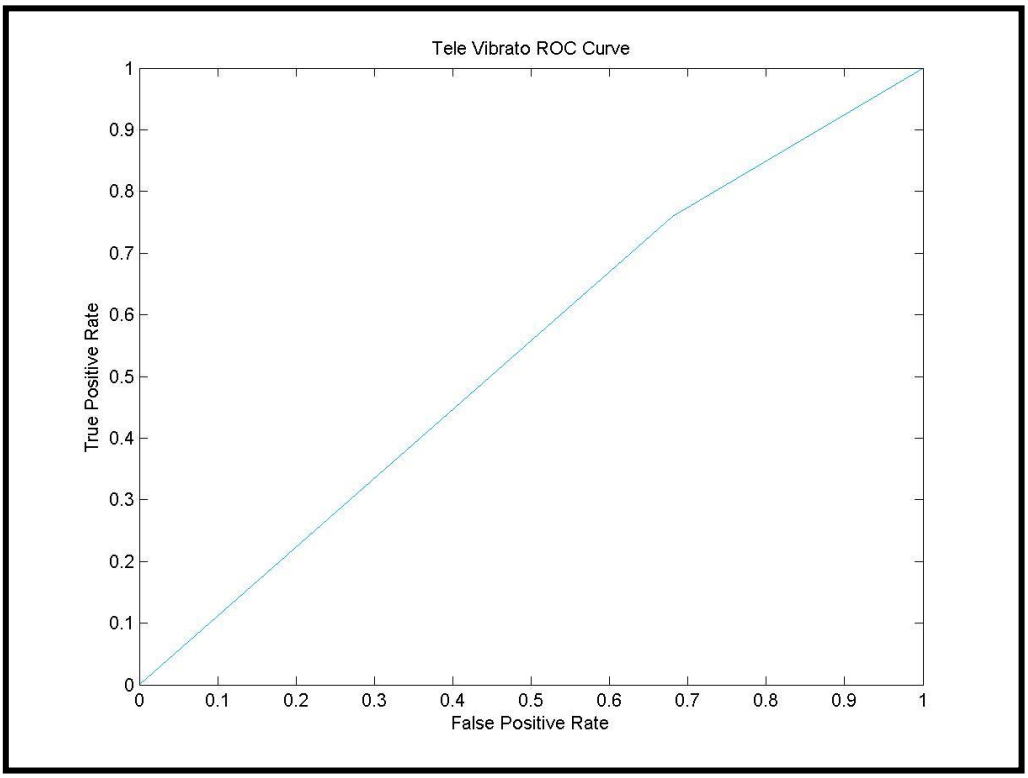

Figure 45 - Tele Vibrato ROC Curve

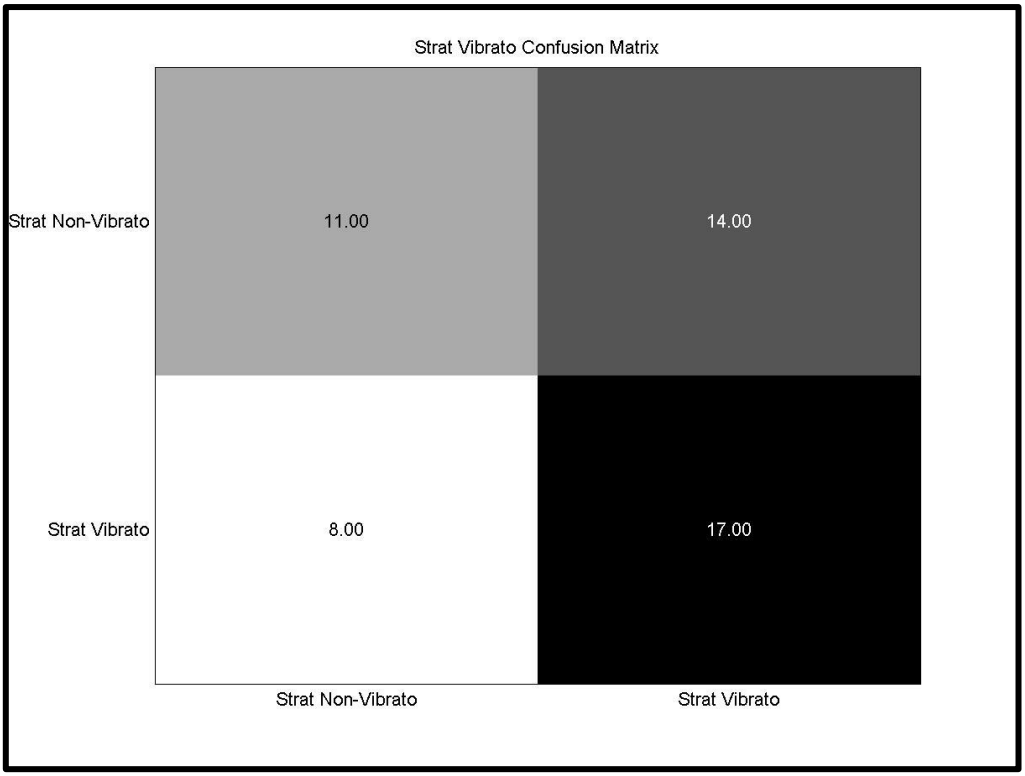

Figure 46 - Strat Vibrato Confusion Matrix

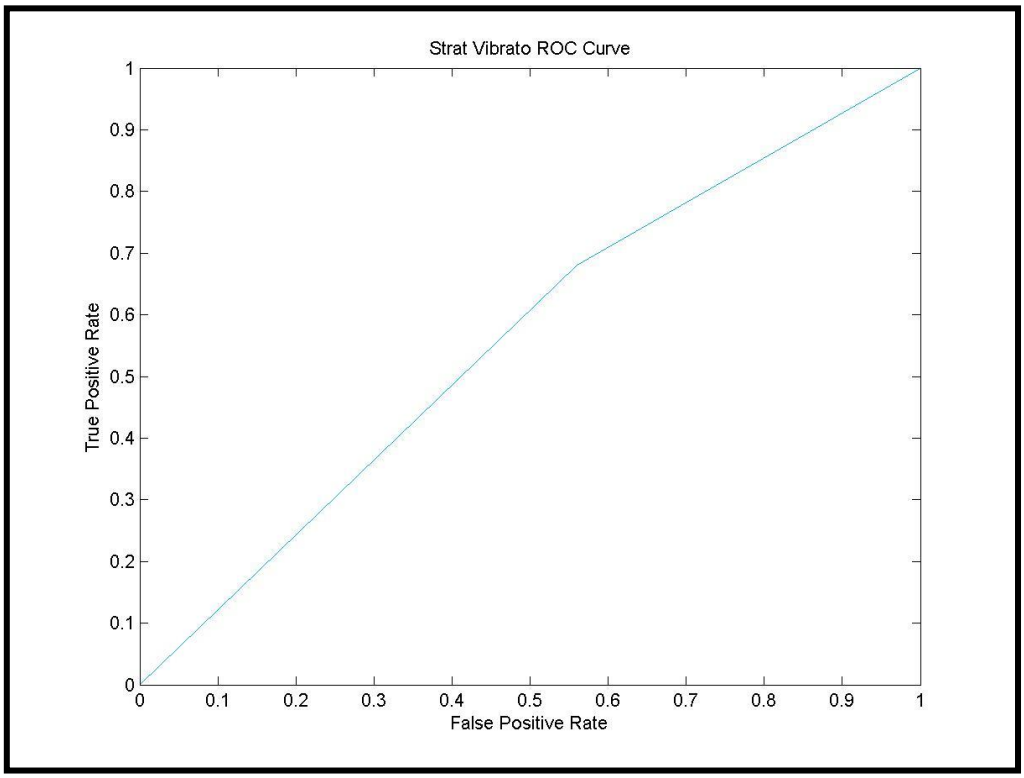

Figure 47 - Strat Vibrato ROC Curve

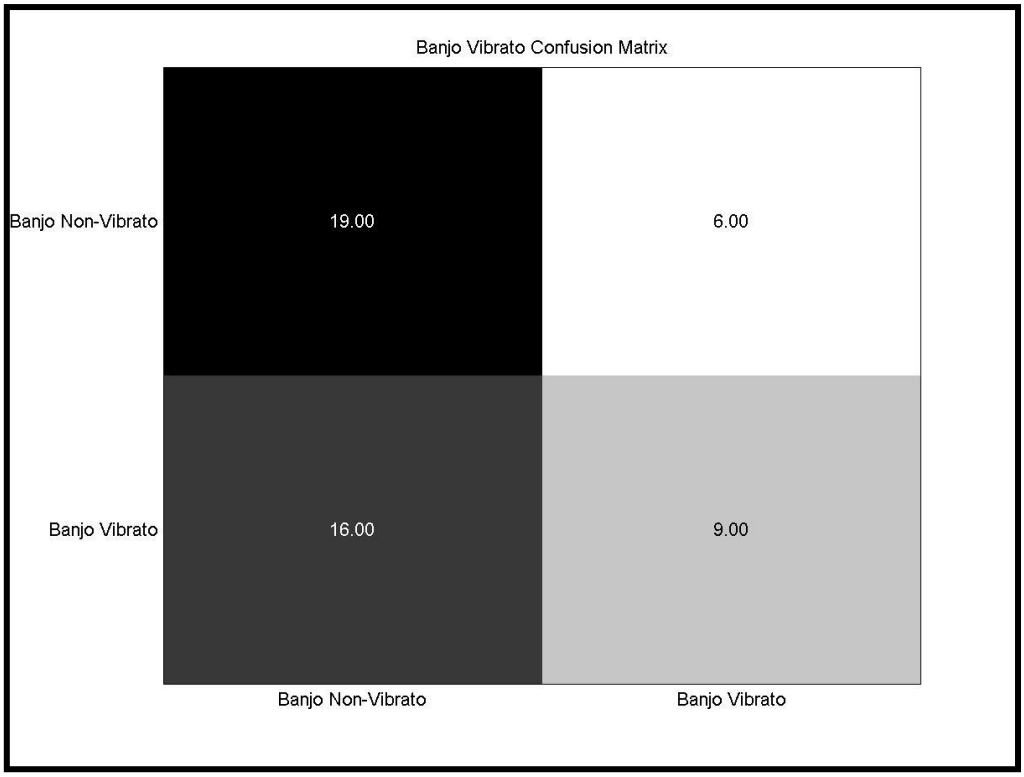

Figure 48 - Banjo Vibrato Confusion Matrix

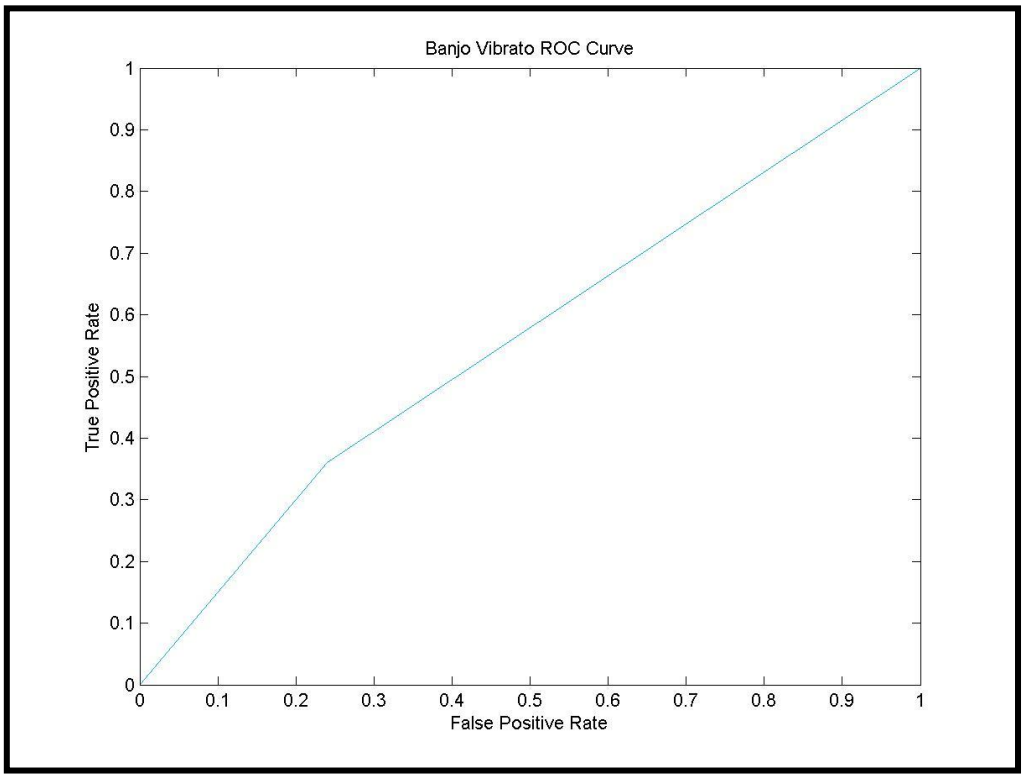

Figure 49 - Banjo Vibrato ROC Curve

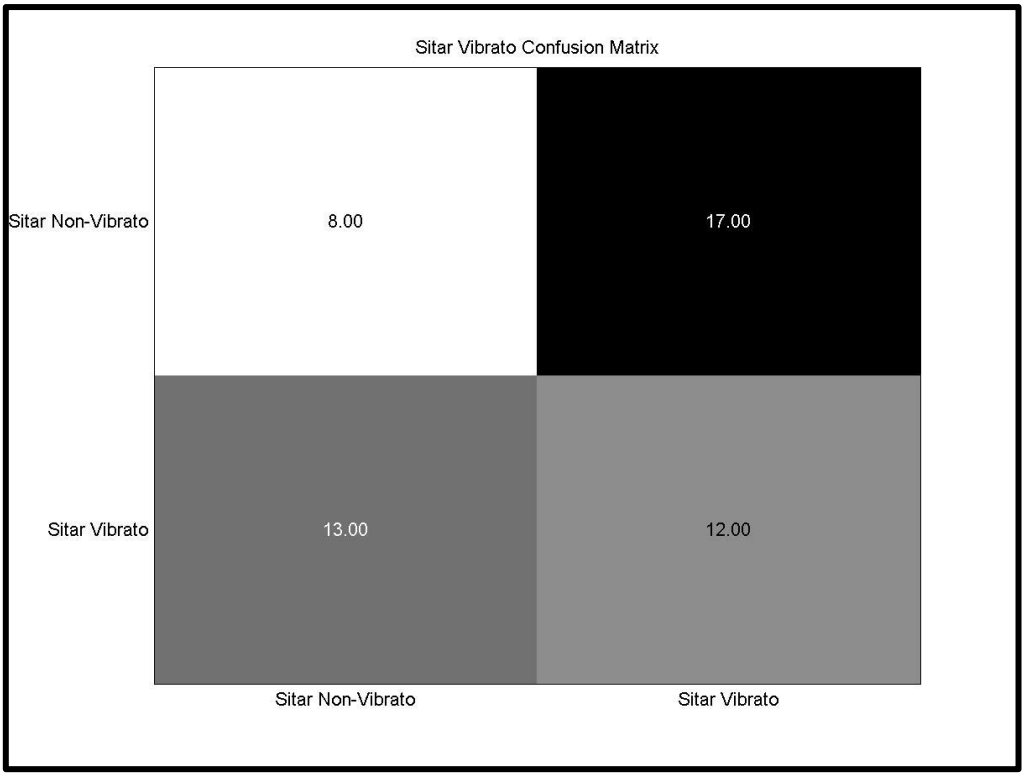

Figure 50 - Sitar Vibrato Confusion Matrix

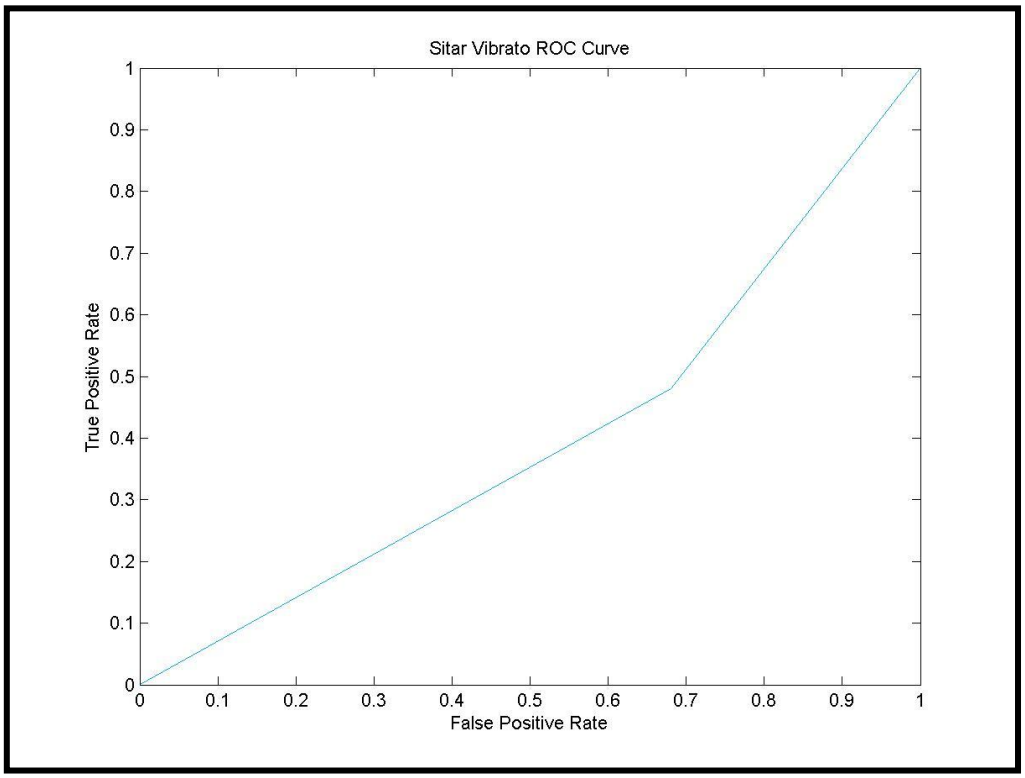

Figure 51 - Sitar Vibrato ROC Curve

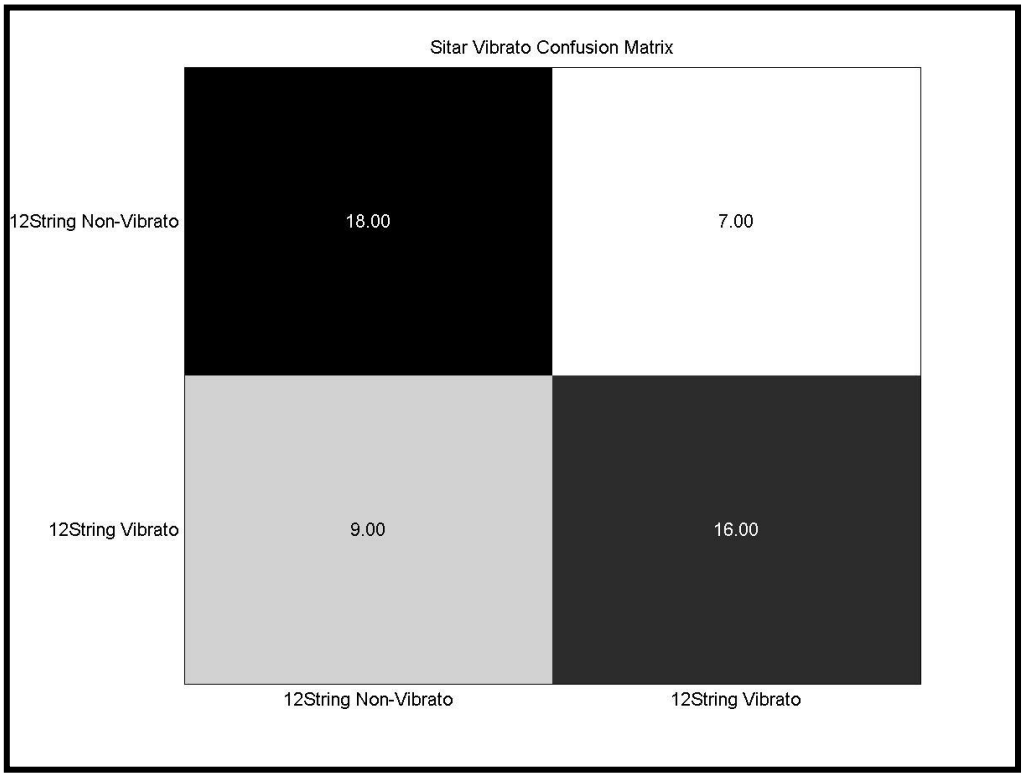

Figure 52 - Sitar Vibrato Confusion Matrix

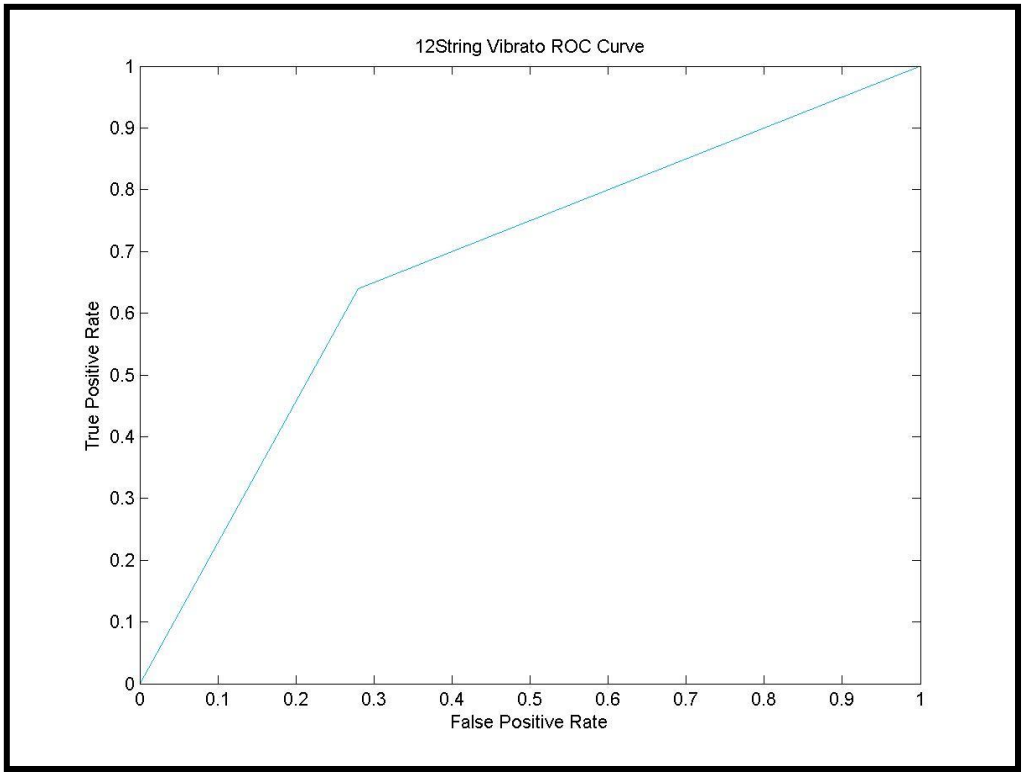

Figure 53 - 12 String Vibrato ROC Curve

D: Single Note Different String Confusion Matrices

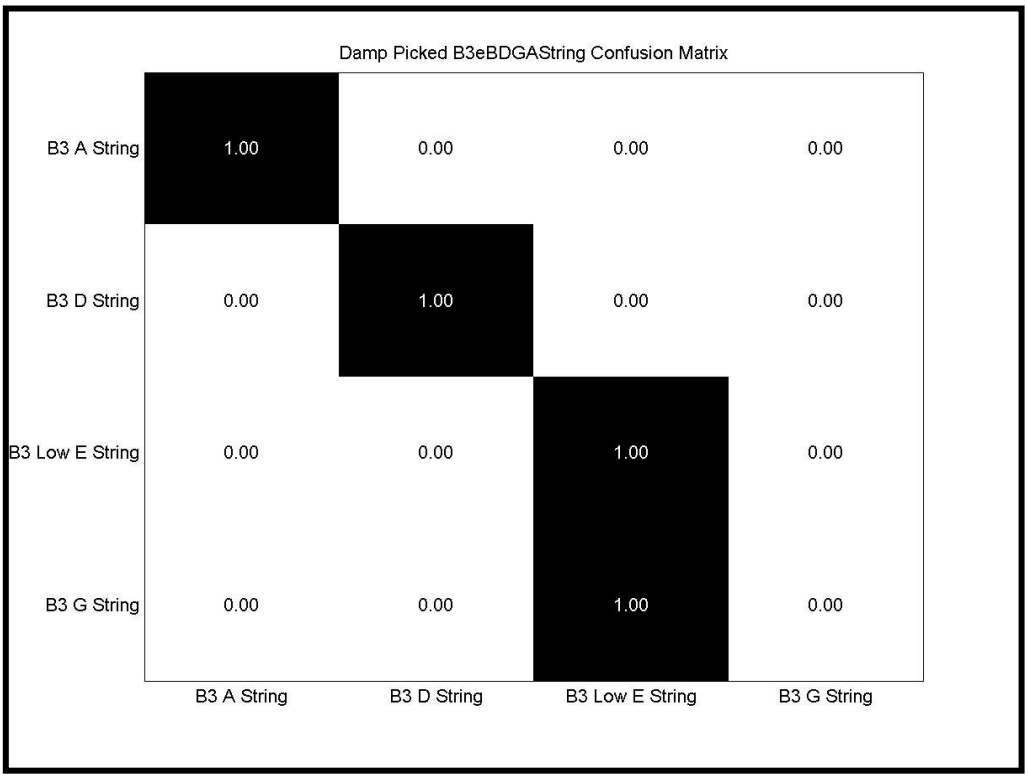

Figure 54 - Damp Picked B3eBDGAString Confusion Matrix

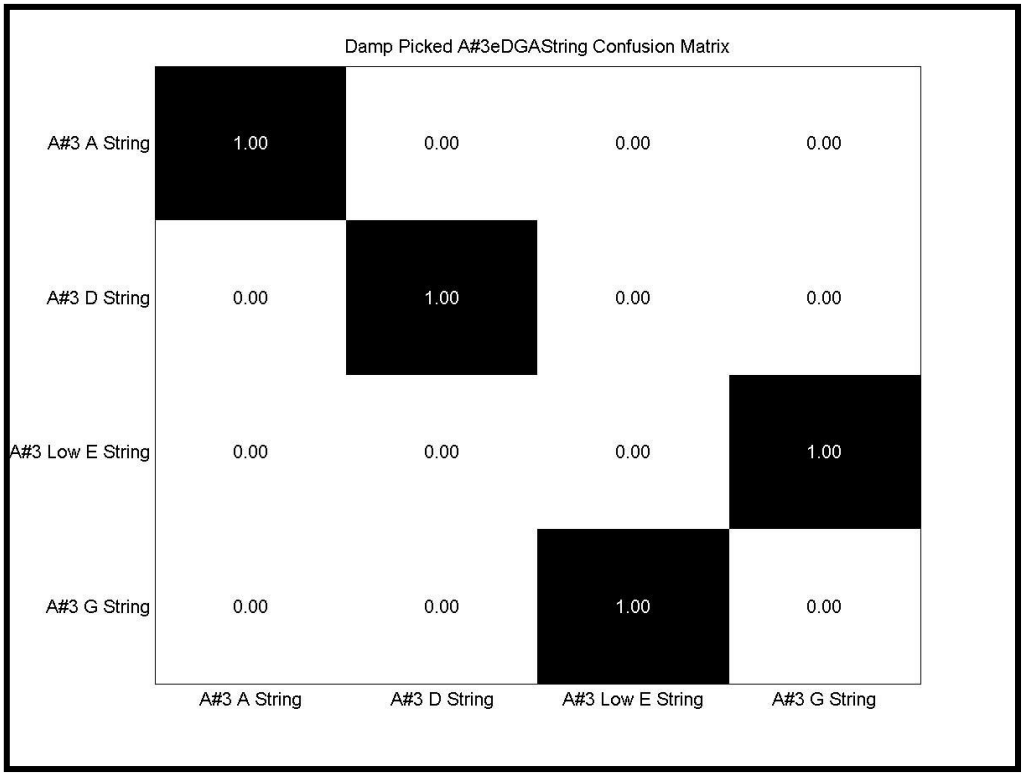

Figure 55 - Damp Picked A#3eDGAString

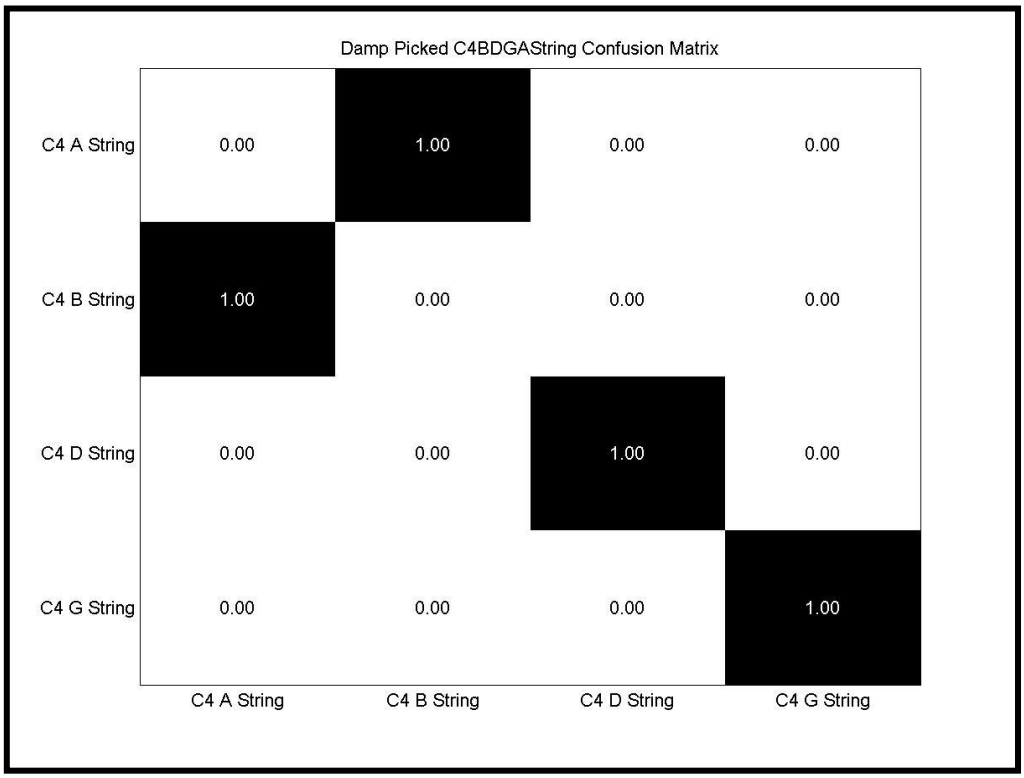

Figure 56 - Damp Picked C4BDGAString Confusion Matrix

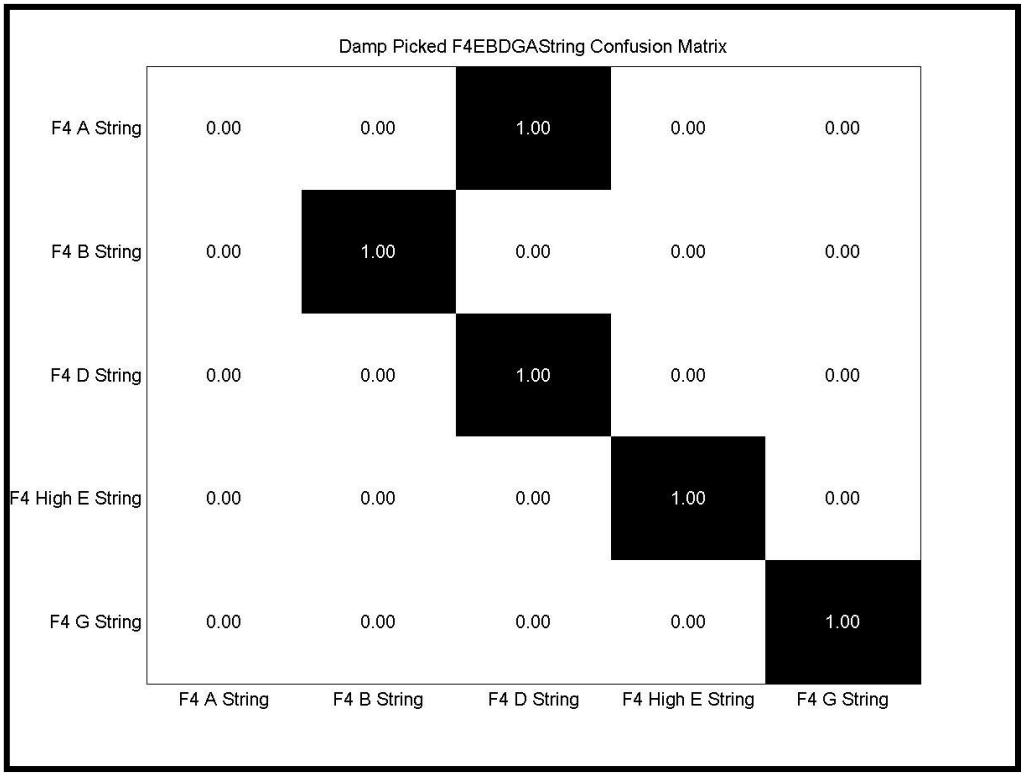

Figure 57 - Damp Picked F4EBDGAString Confusion Matrix

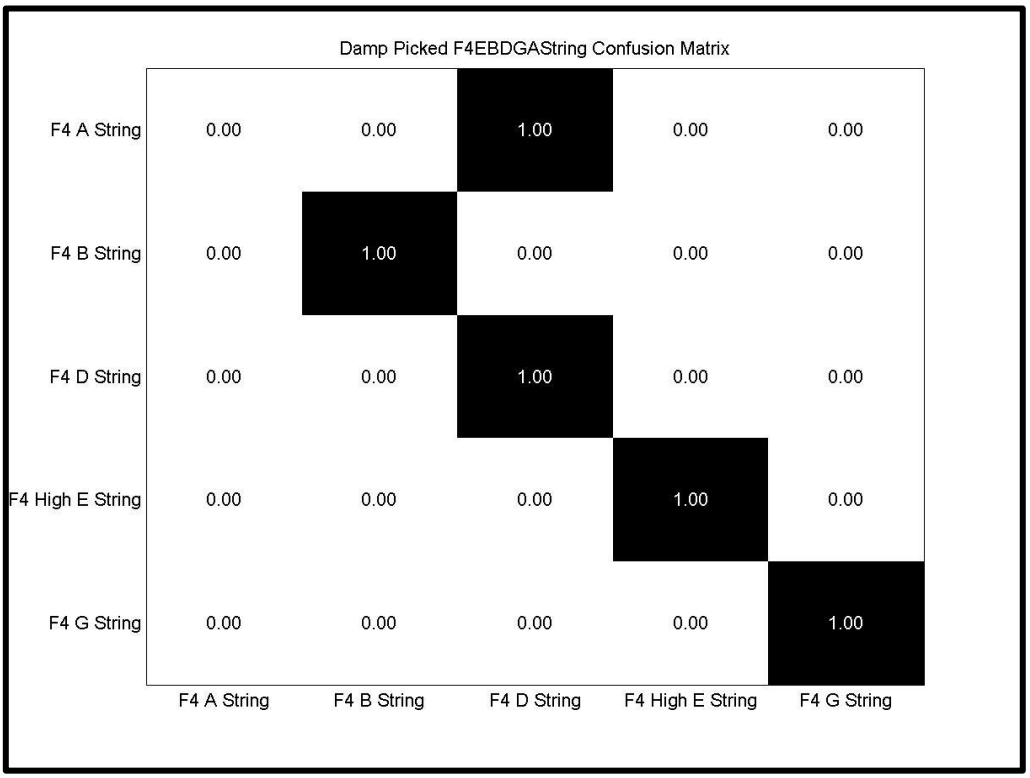

Figure 58 - Damp Picked C5EBGString Test Confusion Matrix

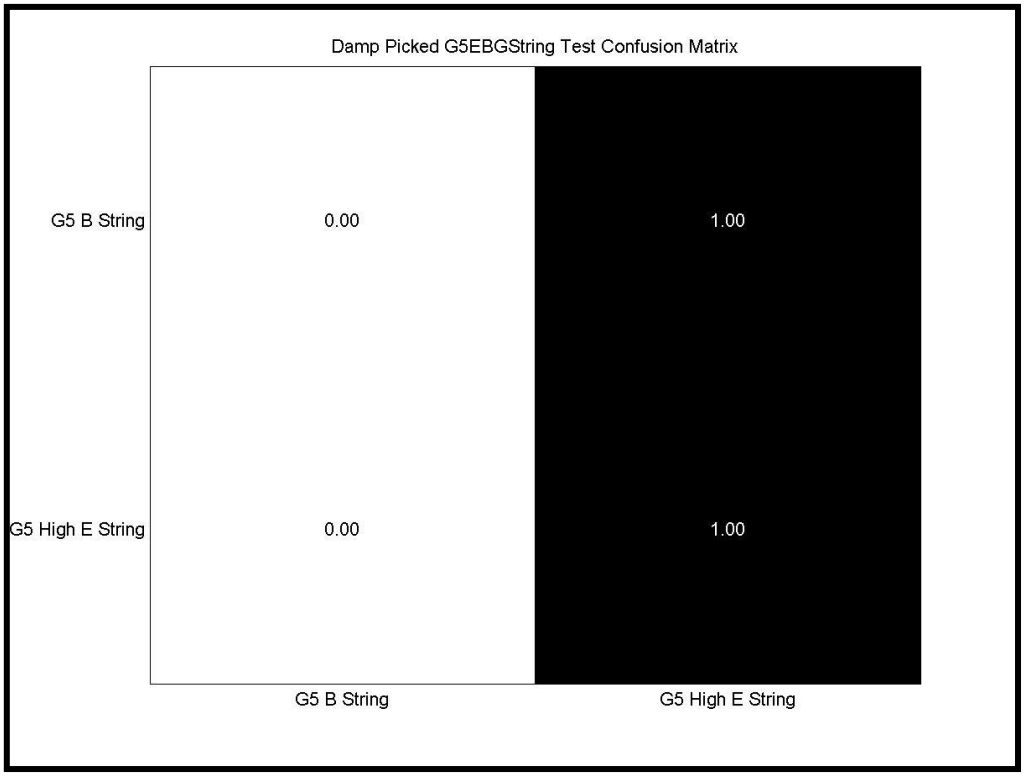

Figure 59 - Damp Picked G5EBGString Confusion Matrix

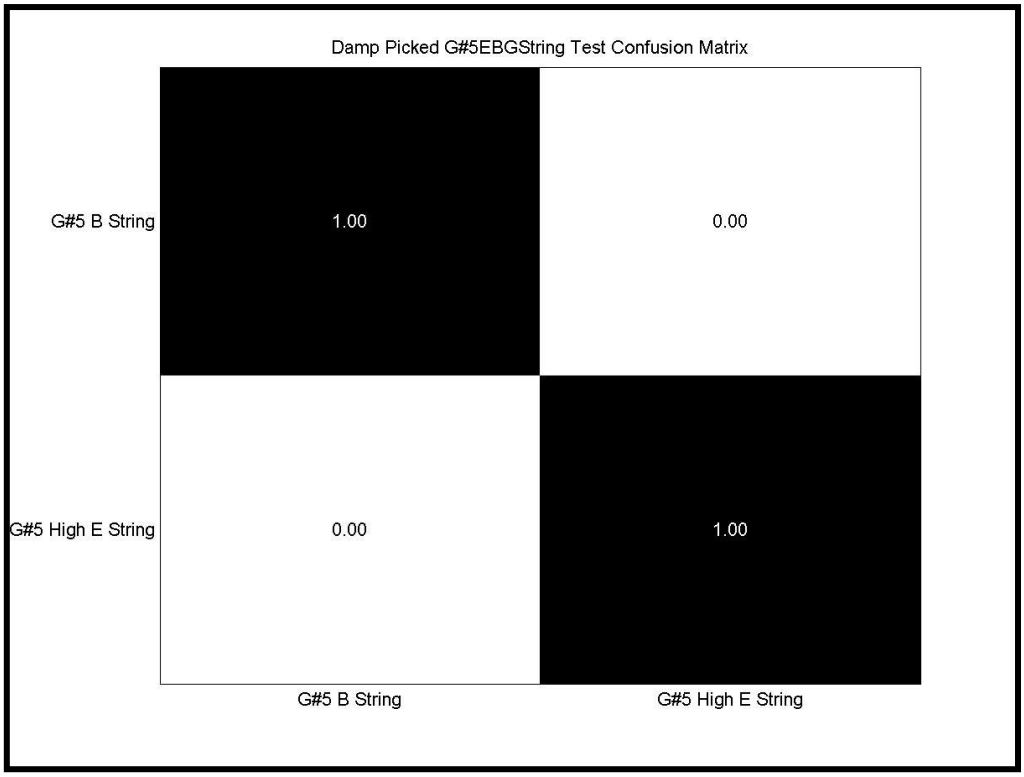

Figure 60 - Damp Picked G#5EBGString Test Confusion Matrix

E: Single Note Same String 12 Frets Confusion Matrices / ROC Curves

E1 – Low E String Confusion Matrix / ROC Curve

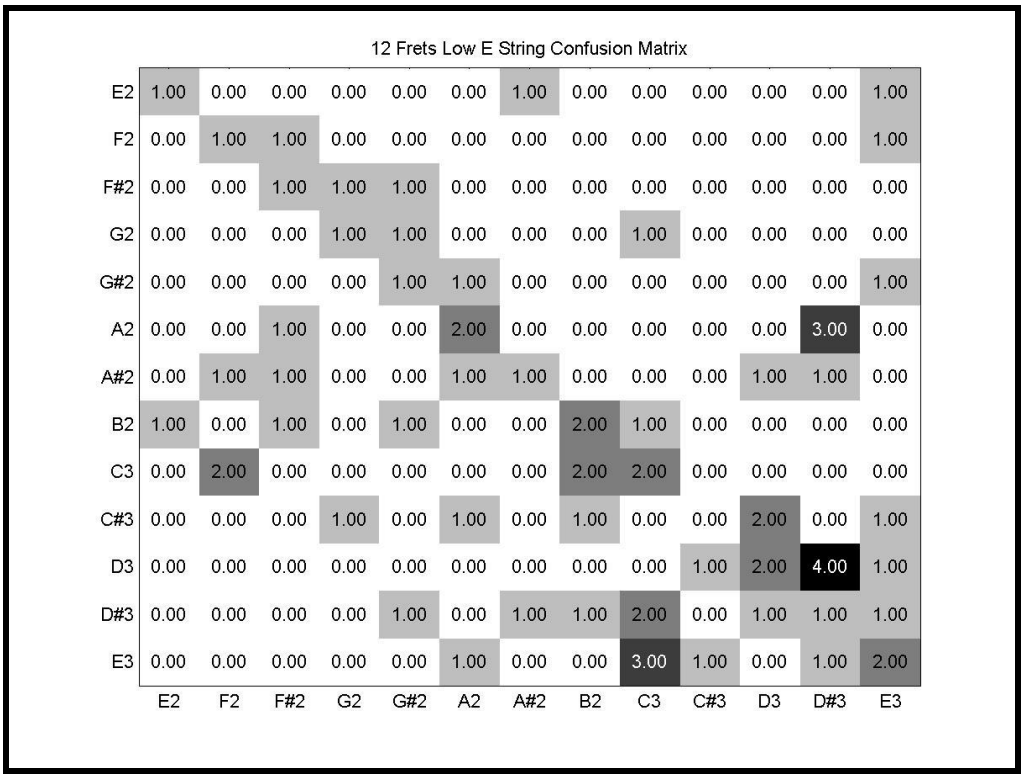

Figure 61 - First 12 Frets Low E Confusion Matrix

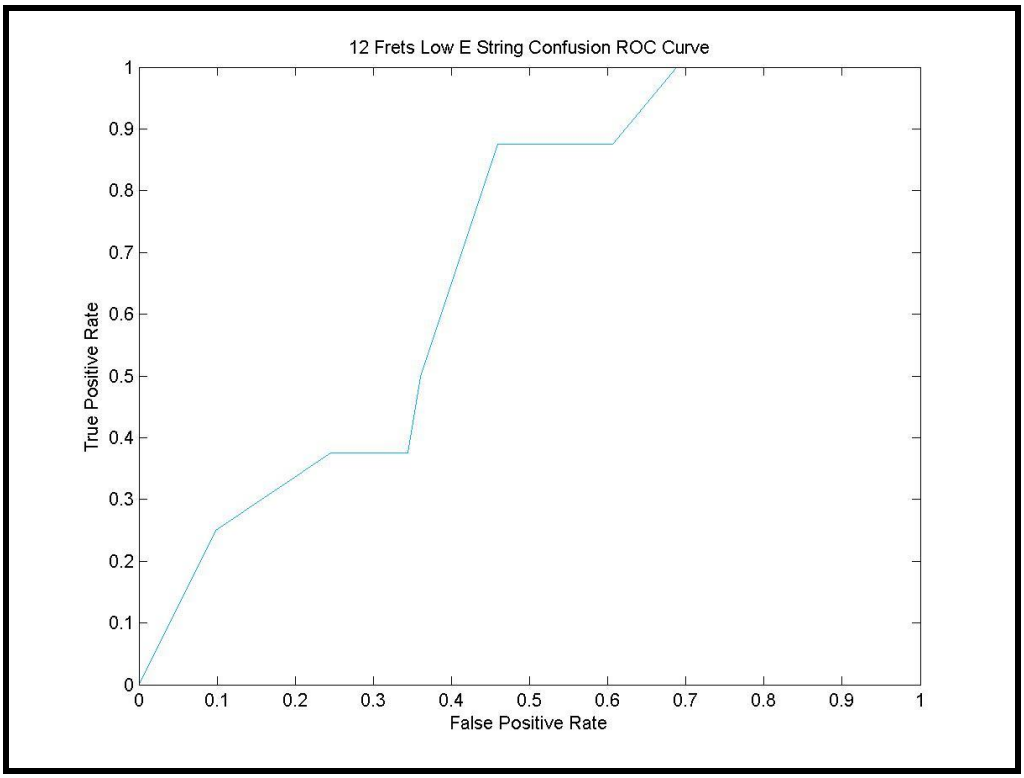

Figure 62 - First 12 Frets Low E ROC Curve

E2 – A String Confusion Matrix / ROC Curve

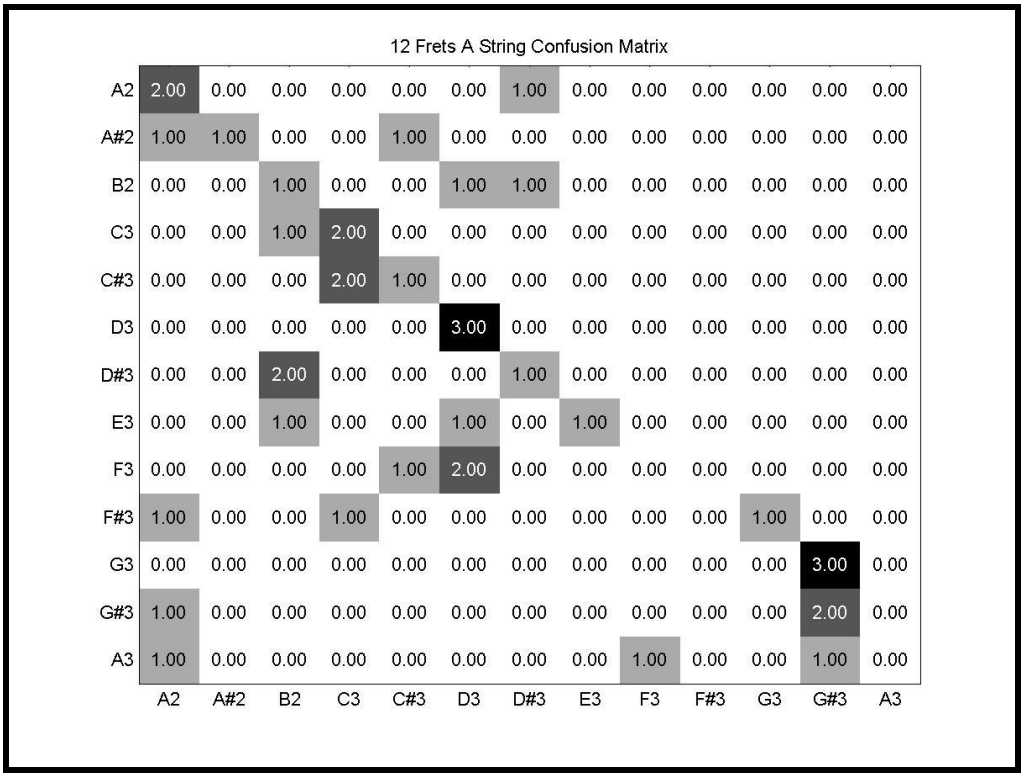

Figure 63 - First 12 Frets A Confusion Matrix

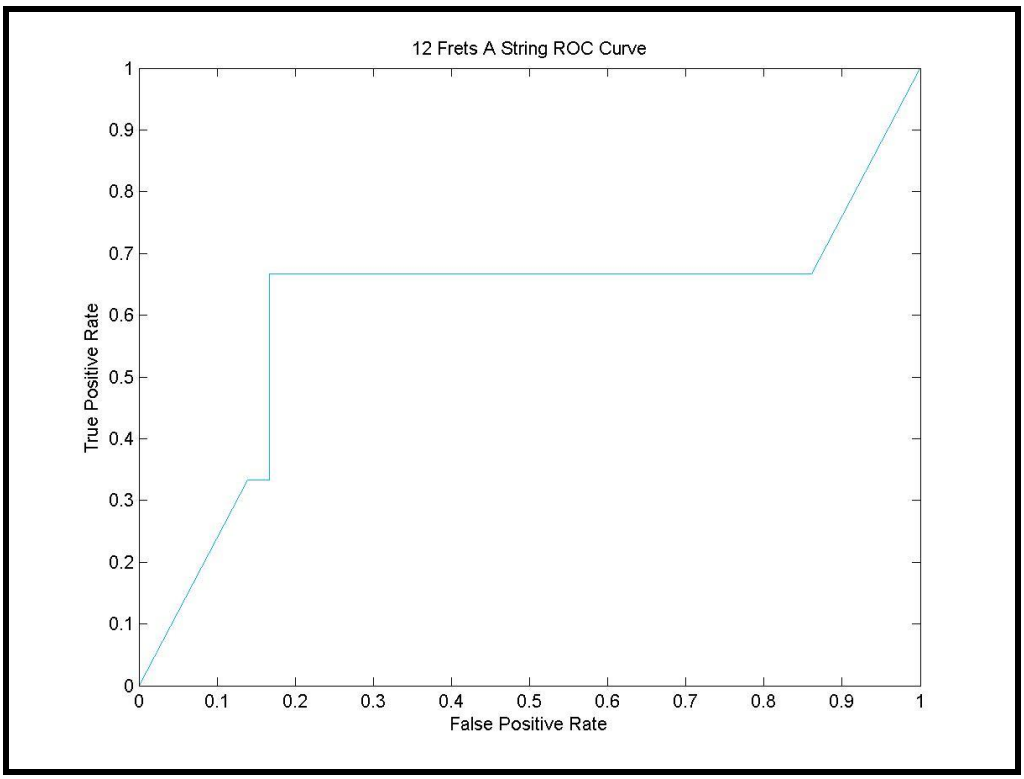

Figure 64 - First 12 Frets A ROC Curve

E3 – D String Confusion Matrix / ROC Curve

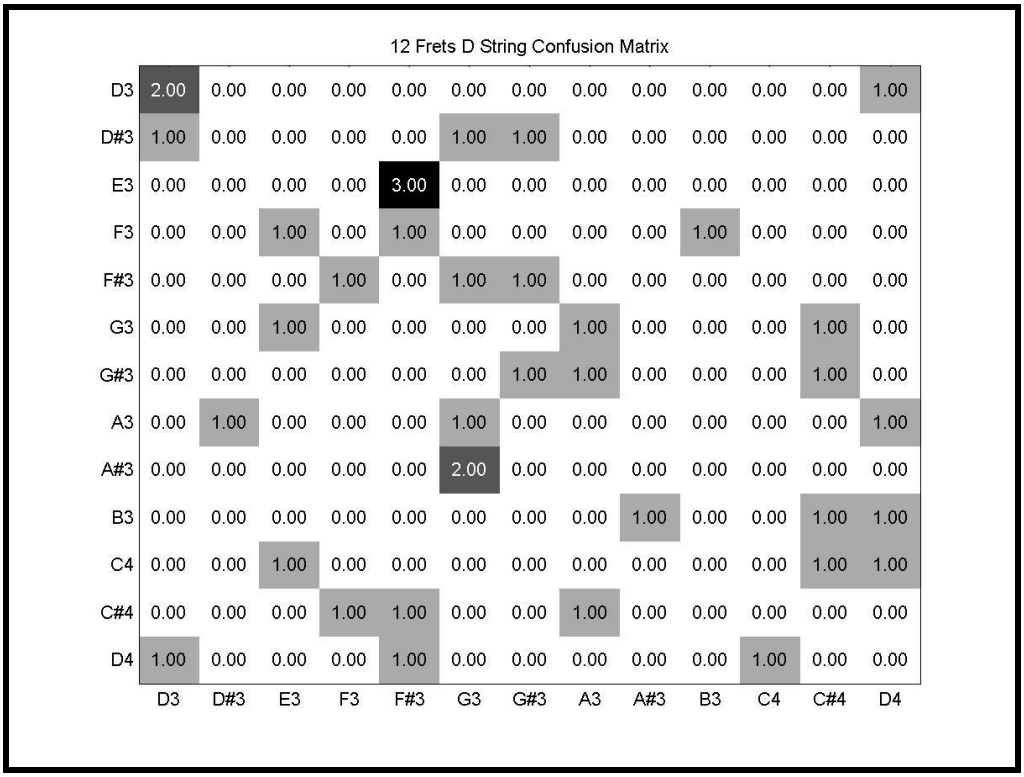

Figure 65 - First 12 Frets D Confusion Matrix

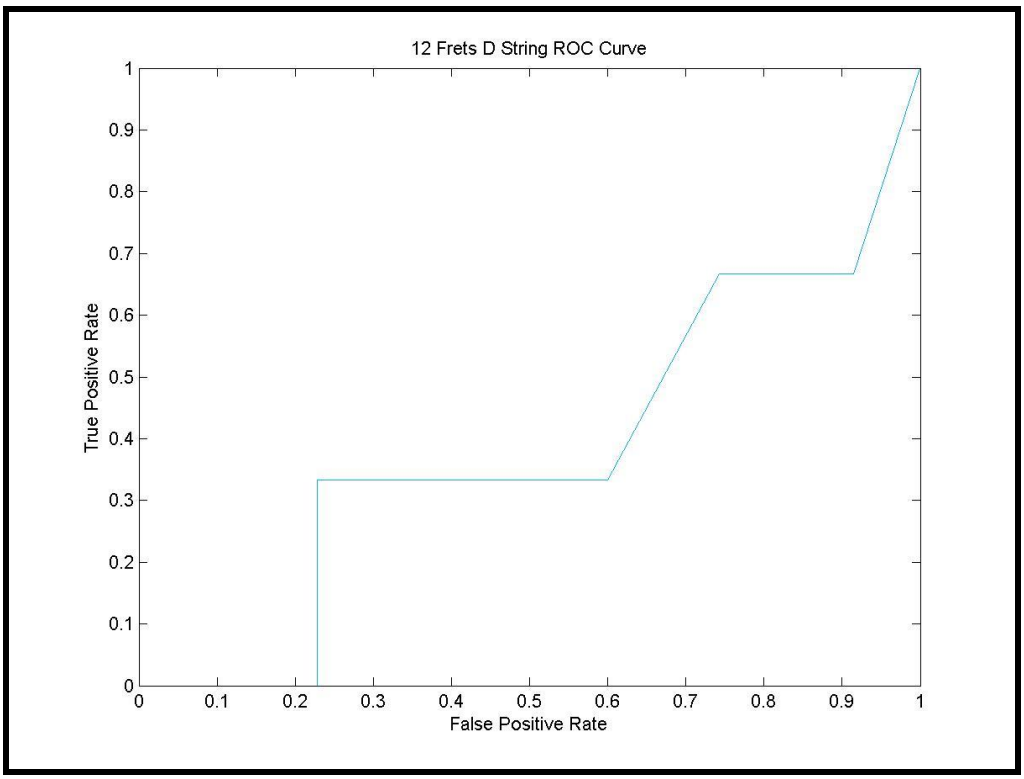

Figure 66 - First 12 Frets D ROC Curve

E4 – G String Confusion Matrix / ROC Curve

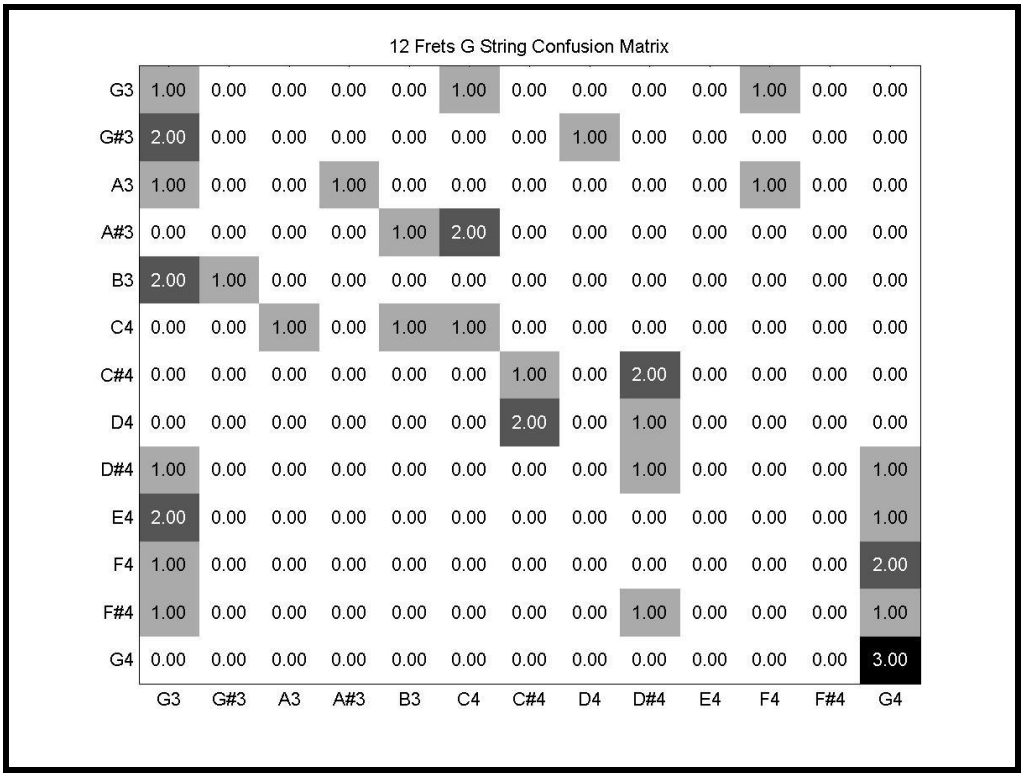

Figure 67 - 12 Frets G String Confusion Matrix

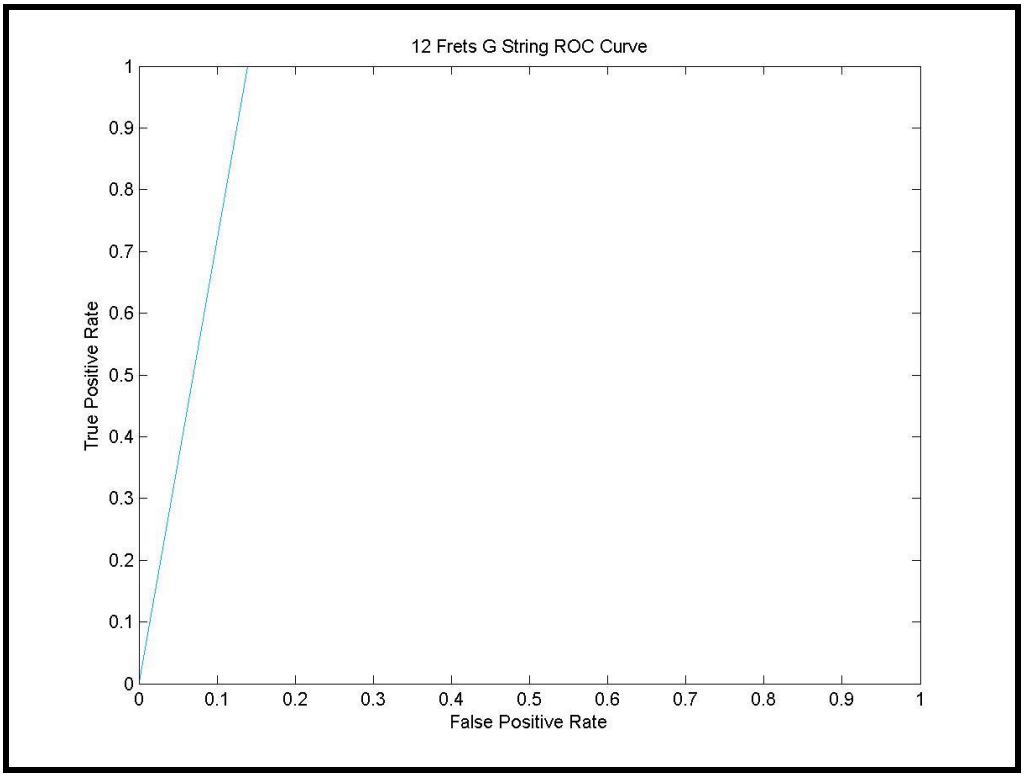

Figure 68 - 12 Frets G String ROC Curve

E5 – B String Confusion Matrix / ROC Curve

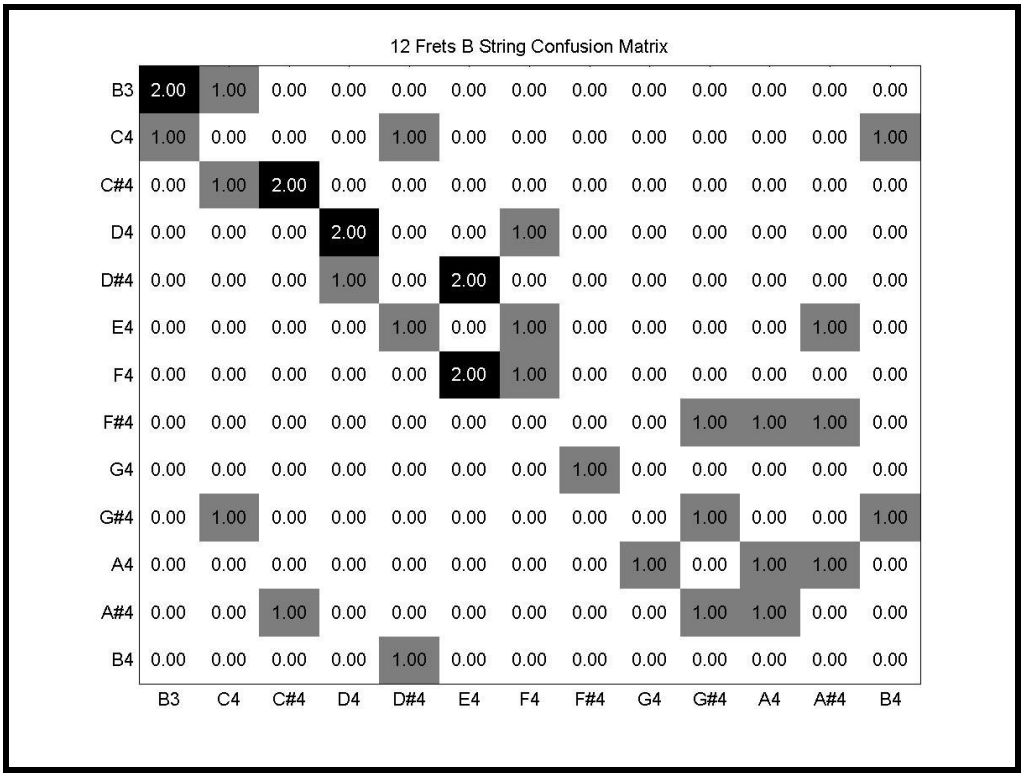

Figure 69 - 12 Frets B String Confusion Matrix

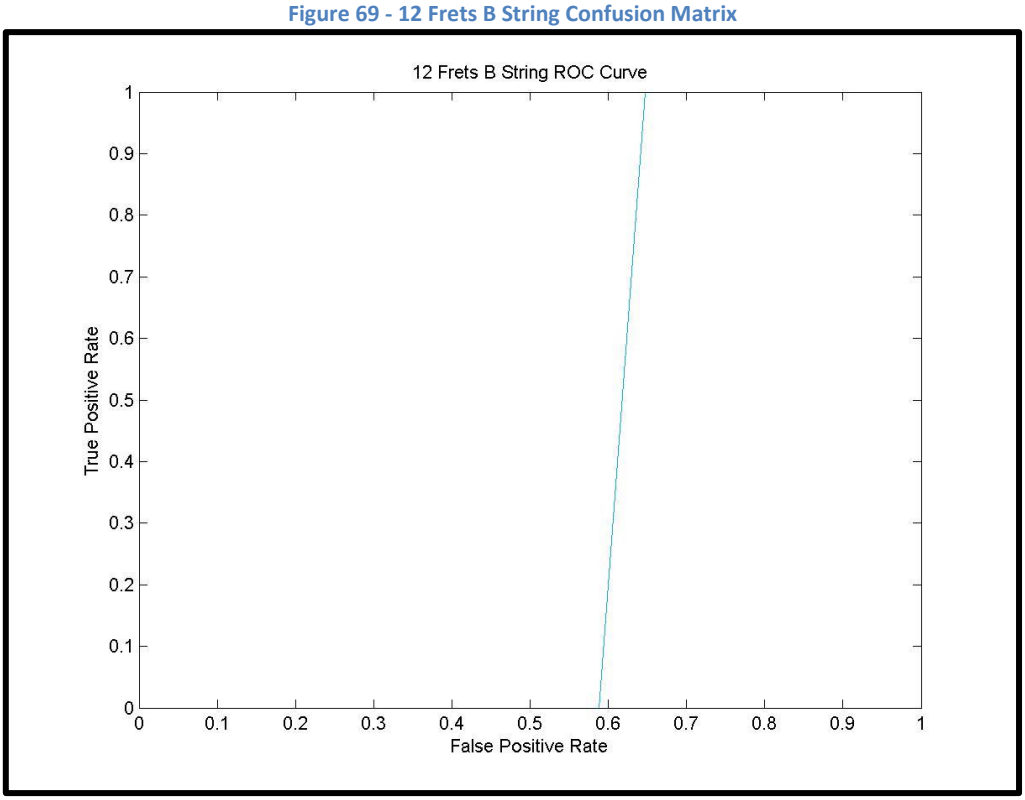

Figure 70 - 12 Frets B String ROC Curve

E6 – High E String Confusion Matrix / ROC Curve

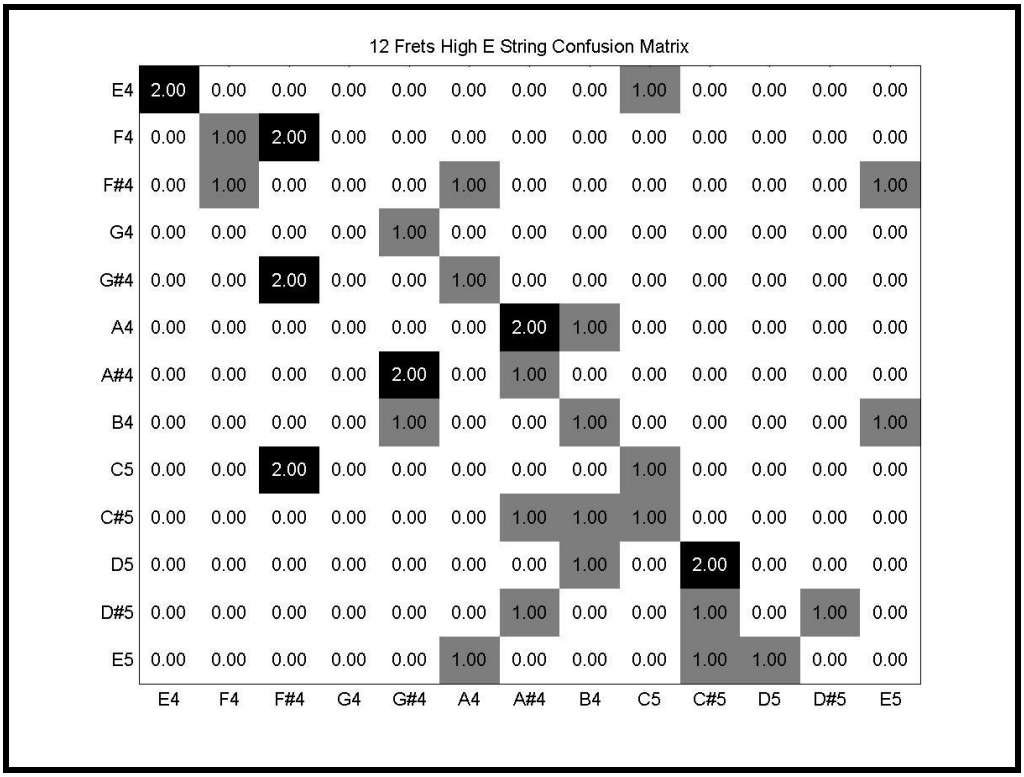

Figure 71 - 12 Frets High E String Confusion Matrix

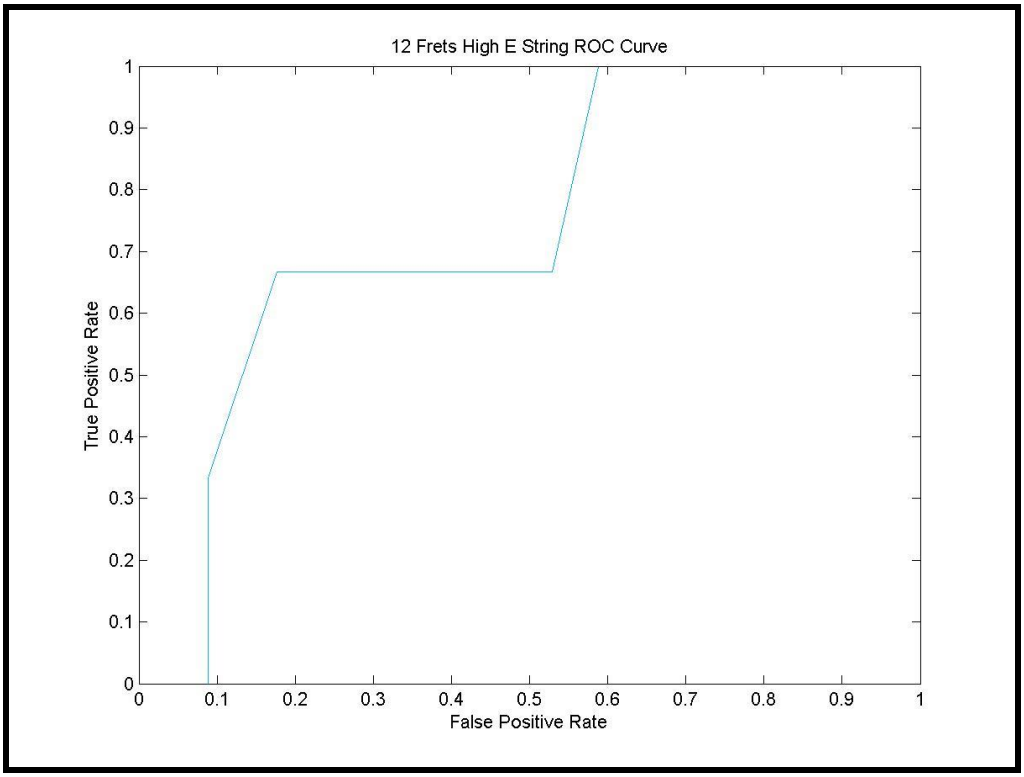

Figure 72 - 12 Frets G String ROC Curve

F – All Notes All String Confusion Matrix