# Machine Learning for Chord Recognition

Reva Abramson
Graduate School of Engineering and Applied Sciences
Columbia University
reva.abramson@columbia.edu / ra2659@columbia.edu

*Abstract*—Due to the advent of the Python audio package Librosa in 2015, automatic chord recognition seems to have become more attainable. In this paper, I discuss a simplified method for recognizing chords with machine learning using datasets of hundreds of human annotated songs along with the corresponding audio files in mp3 or other formats. Apache Spark over Hadoop was used because of its speed of processing large datasets, and therefore the random forest machine learning algorithm used is from the pyspark.mllib package for Python.

***Keywords: big data; machine learning; chords; Librosa; Hadoop, Apache Spark; pyspark; chromagram; pitch class;***

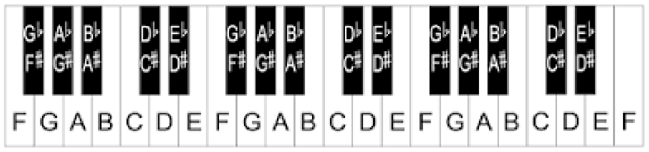
## I. Introduction

Having an ear for music is a genetic gift. For the rest of us, playing songs we like on our instruments can be frustrating. The chords we use are often incorrect. Even professional musicians often disagree on the proper chord. Through machine learning, a program or an app can learn to recognize chords using training examples manually annotated with chords by people with an ear for music. Then, given a new song, the program can give the player the correct chords to play! Automatic chord recognition can also be useful for tuning instruments and for Music Information Retrieval (MIR) which is the study of music.

For this project, I used two datasets containing hundreds of files of annotated (labeled) songs along with the matching audio file. Using Librosa, the Python audio package, I extracted feature vectors from the audio for each time frame in the annotated files and labeled each one with the given chord. The feature vectors and labels were then fed to a multi-class classification algorithm in Spark, and a model was created. I then programmed a common gateway interface (CGI) web application to upload new songs, extract feature vectors using Librosa, and fit the song to the model – resulting in machine learned chord-annotated time frames.

## II. Background

In this paper, I will be referring to the piano, although the ideas apply to most instruments. In Western music, there are twelve *notes*. They are ***C, C# (Db), D, D# (Eb), E, F, F# (Gb), G, G# (Ab), A, A# (Bb), B, B#***. C# is pronounced “C Sharp” and Db is pronounced “D Flat” etc. The black notes have two names depending on which key you’re playing, but they sound the same and their corresponding major and minor chords are the same.



When the *melody* of the first line of the song “Happy Birthday to You” is played on the piano starting from the G note, the notes played are G-G-A-G-C-B; one note for each syllable. A note is a *pitch* which is how high or low a note sounds. A higher note has a higher frequency and a shorter wavelength. E is higher than D which is higher than C [1].

*Chords* are the harmonies. They are groupings of notes. For example, the C major chord is made up of the notes C-E-G and the G major chord is made up of the notes G-B-D.
If someone is singing the line “Happy Birthday to You,” the piano player *accompanies* them by playing two chords on the piano. In other words, each *time frame* in a song has a chord as shown below.

```
C                 G
Happy Birthday to You
```

There are many different types of chords, or combination of notes. For this project I only classify basic major and minor chords. Each note has both a major and minor chord.

| Example Chord | Notes |
|---|---|
| C major | C-E-G |
| C minor | C-Eb-G |

To play a song on the piano, the player needs to know the notes for the melody along with the chords for the harmony.

To accompany a singer, the player needs the chords. It's not too difficult with some training to play a melody – it's about listening to whether a tune goes up or down. On the other hand, knowing the chords to play is very difficult without an ear for music.

## III. Related Works

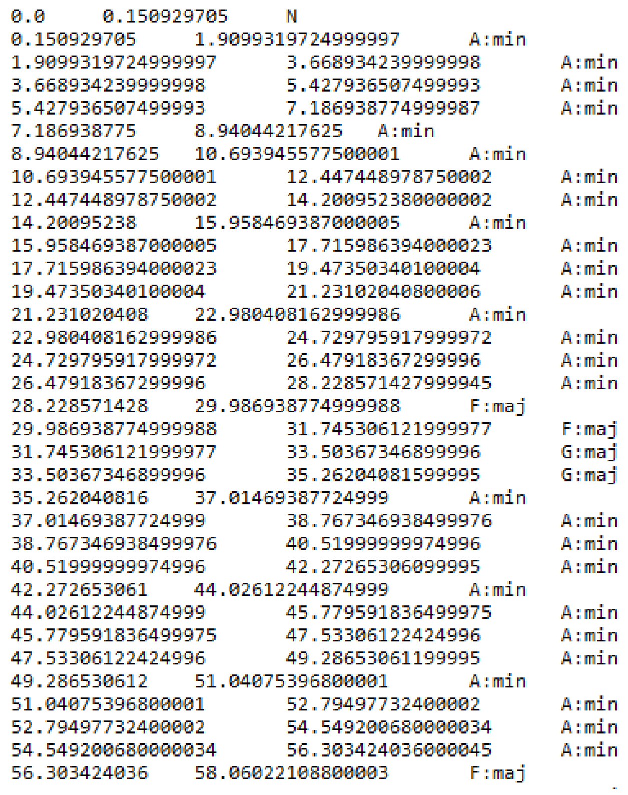
Starting in 1999, there was quite a bit of work related to extracting chords from audio. All this research eventually led to the 2015 creation of the audio python library Librosa in 2015 which was used extensively for this project. Below is a table of related work to this project summarized from the doctoral thesis of Matthew McVicar [2].

| Year | Contributor | Relevant Contribution |
|---|---|---|
| 1999 | G.H. Wakefield | Mathematics of chromagram feature vectors |
| 2000 | J.P. Bello et al. | Methods for tracking pitch |
| 2001 | S.H. Nawab et al. | Use of Constant-Q Spectrum |
| 2002 | C. Raphael | HMM (Hidden Markov Model) for melody extraction |
| 2003 | A.Sheh, D. Ellis | HMM for chord extraction and training from labeled data |
| 2004 | S. Pauws | Removal of background spectrum and extracting the key |
| 2006 | E. Gomez P. Herrera | Song identification using chromagrams |
| 2009 | T. Cho, J.P. Bello | Real time chord recognition |
| 2009 | J.T. Reed | Harmonic and Purcussive Source Separation (HPSS) |
| 2011 | M. McVicar | Using Dynamic Bayesian Networks for chord extraction |
| 2015 | B. McFee | LIBROSA |

There was a lot of work done using *Hidden Markov Models* (HMMs) for chord processing because the state of the previous chords helps determine the current chord. However, I found no HMM capability for multiple classes in either Spark or Hadoop.

## IV. System Overview

Two datasets were used containing hundreds of files of annotated songs. Both datasets have similar formats, so any inconsistencies were adapted by the Python code. The image below is the annotated file of "She Bop" from a 1984 album by Cindi Lauper. The song is divided into short time frames and each time frame is *annotated* or *labeled* by a chord. The first column is the start time (of the time frame), the second column is the end time (of the time frame), and the third column is the chord.



```
0.0       0.150929705       N
0.150929705       1.9099319724999997        A:min
1.9099319724999997        3.668934239999998         A:min
3.668934239999998         5.427936507499993         A:min
5.427936507499993         7.186938774999987         A:min
7.186938775       8.94044217625    A:min
8.94044217625     10.693945577500001        A:min
10.693945577500001        12.447448978750002        A:min
12.447448978750002        14.200952380000002        A:min
14.20095238       15.958469387000005        A:min
15.958469387000005        17.715986394000023        A:min
17.715986394000023        19.47350340100004         A:min
19.47350340100004         21.23102040800006         A:min
21.231020408      22.980408162999986        A:min
22.980408162999986        24.729795917999972        A:min
24.729795917999972        26.47918367299996         A:min
26.47918367299996         28.228571427999945        A:min
28.228571428      29.986938774999988        F:maj
29.986938774999988        31.745306121999977        F:maj
31.745306121999977        33.50367346899996         G:maj
33.50367346899996         35.26204081599995         G:maj
35.262040816      37.01469387724999         A:min
37.01469387724999         38.767346938499976        A:min
38.767346938499976        40.51999999974996         A:min
40.51999999974996         42.27265306099995         A:min
42.272653061      44.02612244874999         A:min
44.02612244874999         45.779591836499975        A:min
45.779591836499975        47.53306122424996         A:min
47.53306122424996         49.28653061199995         A:min
49.286530612      51.04075396800001         A:min
51.04075396800001         52.79497732400002         A:min
52.79497732400002         54.549200680000034        A:min
54.549200680000034        56.303424036000045        A:min
56.303424036      58.06022108800003         F:maj
```

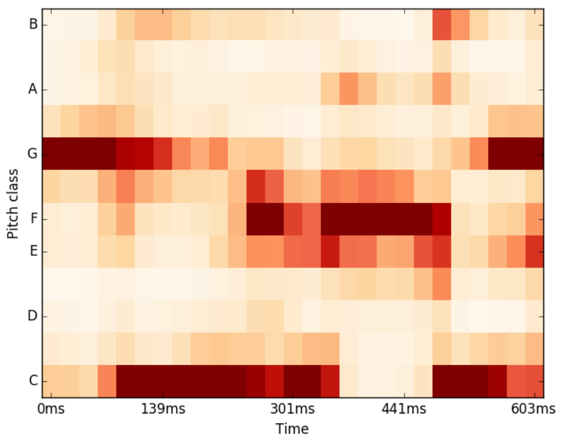
The first dataset is the Isophonics dataset. It contains many annotated albums from Michael Jackson, Queen, Carole King, and the Beatles. It is from the Centre for Digital Music (C4DM) at Queen Mary, University of London [3].

The second dataset is the Billboard 100 dataset. It contains hundreds of songs that were on the Billboard Hot 100 since the 1950s. It is from the McGill Billboard project at McGill University [4].

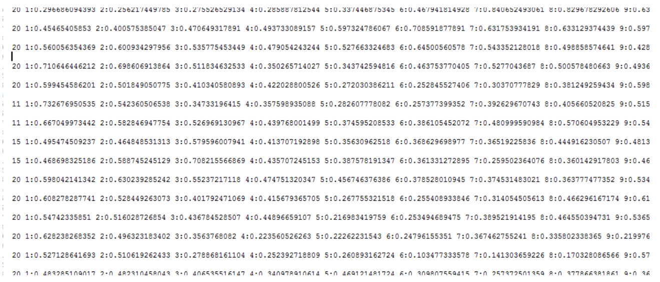
Each annotated file corresponds to an audio file from the original album. These had to be retrieved separately. Because of copyright issues they were not paired with the files[4]. It was a manual effort to search youtube for the music to match the files. It involved ensuring that the youtube video was not an adaptation of the original song, and additionally, matching up the lengths of the video with the final time frame value in the file. Once the youtube videos were located, I converted them into mp3 format using the free site convert2mp3.net [5].

The songs I ended up using the most were from the Isophonics Beatles dataset because those were manually annotated and thoroughly checked by *Music Information Retrieval (MIR)* professionals [6]. For simplicity, my code ignored any chords that are not in the basic set of 24 major and minor chords plus 1 for the "No Chord."

My system is divided into 3 parts. The first Python program uses Librosa to extract feature vectors from the dataset. Each time frame record is converted into 12 features - one for each note - and a class label which is the chord. The chord is converted to a numeric value to be used in arrays with the numeric features by the algorithms. There are 12 features because there are 12 notes in Western Music. Each feature value is a decimal number between 0 and 1. It is an average of how much of that note is heard in the time frame. Since the basic chords are made up of 3 notes, those notes should have the highest values in the time frame even though other notes may be heard as well. We can see this pictorially in the *chromagram* below. A chromagram shows how much of each note is heard (y axis) for every millisecond in the timeframe (x axis) [7]. The image below is clearly a C major chord because of the strength of the color for the notes C, E, and G.



The second part takes in the output from the first part as shown below. Each training example is on its own row with a class label and 12 feature ids and values (only 9 shown).



```
20 1:0.296686094393 2:0.256217449785 3:0.275526529134 4:0.285887812544 5:0.337446875345 6:0.467941814928 7:0.840652493061 8:0.829678292606 9:0.63
20 1:0.45465405853 2:0.400575385047 3:0.470649317891 4:0.493733089157 5:0.597324786067 6:0.708591877891 7:0.631753934191 8:0.633129374439 9:0.597
20 1:0.560056354369 2:0.600934297956 3:0.535775453449 4:0.479054243244 5:0.527663324683 6:0.64500560578 7:0.543352128018 8:0.498858574641 9:0.428
20 1:0.710646446212 2:0.698606913864 3:0.511834632533 4:0.350265714027 5:0.343742594816 6:0.463753770405 7:0.5277043687 8:0.500578480663 9:0.4936
20 1:0.599454586201 2:0.501849050775 3:0.410340580893 4:0.422028800526 5:0.272030386211 6:0.252845527406 7:0.30370777829 8:0.381249259434 9:0.598
11 1:0.732676950535 2:0.542360506538 3:0.34733196415 4:0.357598935088 5:0.282607778082 6:0.257377399352 7:0.392629670743 8:0.405660520825 9:0.515
11 1:0.667049973442 2:0.582846947754 3:0.526969130967 4:0.439768001499 5:0.374595208533 6:0.386105452072 7:0.480999590984 8:0.570604953229 9:0.54
15 1:0.495474509237 2:0.464848531313 3:0.579596007941 4:0.413707192898 5:0.35630962518 6:0.368629698977 7:0.36519225836 8:0.444916230507 9:0.4813
15 1:0.468698325186 2:0.588745245129 3:0.708215566869 4:0.435707245153 5:0.387578191347 6:0.361331272895 7:0.259502364076 8:0.360142917803 9:0.46
20 1:0.598042141342 2:0.630239285242 3:0.55237217118 4:0.474751320347 5:0.456746376386 6:0.378528010945 7:0.374531483021 8:0.363777477352 9:0.534
20 1:0.608278287741 2:0.528449263073 3:0.401792471069 4:0.415679365705 5:0.267755321518 6:0.255408933846 7:0.314054505613 8:0.466296167174 9:0.61
20 1:0.54742335851 2:0.516028726854 3:0.436784528507 4:0.44896659107 5:0.216983419759 6:0.253494689475 7:0.389521914195 8:0.464550394731 9:0.5365
20 1:0.628238268352 2:0.496323183402 3:0.3563768082 4:0.223560526263 5:0.2226231543 6:0.24796155351 7:0.367462755241 8:0.335802338365 9:0.219976
20 1:0.527128641693 2:0.510619262433 3:0.278868161104 4:0.252392718809 5:0.260893162724 6:0.103477333578 7:0.141303659226 8:0.170328086566 9:0.57
20 1:0.483285109017 2:0.482310458043 3:0.406535516147 4:0.340978910614 5:0.469121481724 6:0.309807559415 7:0.257372501359 8:0.377866381861 9:0.36
```

Then Spark's random forest multi-class classifier trains on 70% of the data and predicts on 30% of the data. The accuracy is around 70% without any enhancements to the features or the algorithm. The model is saved for future predictions.

The third part of the project is the backend for predicting new songs. It is a combination of the first and second parts. The new song is divided into equal time frames or segments. Librosa extracts each specified time frame and creates a chromagram which become the features. Then using the saved model, Spark predicts chord labels for each segment.

## V. ALGORITHM

*Part i*: In order to assign a required numeric value to each chord label (such as C#:maj), I created two lists of chords as follows:

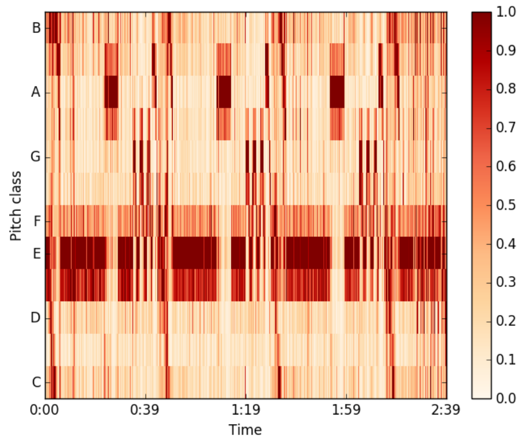
```
["N","C:maj","C:min","C#:maj","C#:min","D:maj","D:min",
"D#:maj","D#:min","E:maj","E:min","F:maj","F:min","F#:maj",
"F#:min","G:maj","G:min","G#:maj","G#min","A:maj","A:min",
"A#:maj", "A#:min", "B:maj","B:min"]
["N","C:maj","C:min","Db:maj","Db:min","D:maj","D:min",
"Eb:maj","Eb:min","E:maj","E:min","F:maj","F:min","Gb:maj",
"Gb:min","G:maj","G:min","Ab:maj","Ab:min","A:maj","A:min",
"Bb:maj","Bb:min","B:maj","B:min"]
```

The index of the chord label became the numeric value of the class. Two lists are required because some chords have different names or labels but the underlying notes are exactly the same. For example, the C# major chord has the same three notes as the Db minor chord and must be treated as one. Therefore, these two chords now have the same index in the lists, and they have the same numeric class value. This reduces the number of classes from 35 to 25 which can improve accuracy.

To create feature vectors I used the python audio library Librosa. To load a time frame of audio for processing, I specified the start time and duration and *sampling rate* which is the number of samples of audio carried per second. The start time and duration are taken from the datasets and the sampling rate is changed based on performance.

For each time frame, I used the *Harmonic Percussive Sound Separation* method from Librosa. This takes out the beats from the sample and leaves the notes in order to get more accurate chromagrams.

Then I used Librosa's Short Term Fourier Transform method to measure pitch or frequency. It assigns a number between 0 and 1 to each note or pitch depending on frequency at a specific moment [8]. Higher notes have higher frequencies and shorter wavelengths. For example, if the sampling rate is 10 times per second and there are 2 seconds in the time frame, then each of the 12 notes (from C to B) get 20 numbers. This *chromagram* can be viewed pictorially using matplotlib in python:



I decided to use each note as one feature, and therefore took an average from all the samples in the time frame. The output of all the training examples and feature vectors and numeric chord labels are shown in the previous section. To use Librosa, I also installed ffmpeg in order for Librosa to be able to process different formats of audio [9].

*Part ii*: I used Apache Spark to train on the examples and create the model [10]. Spark can either use *Resilient Distributed Datasets* or *Data Frames* for the data structures behind the scenes [11]. I found it necessary to use the RDD methods in order to save the model and not have to retrain on the labeled example dataset with every new song which would be impossibly slow.

I used Spark's Random Forest Classifier algorithm because it's one of the only algorithms in current Spark that could do well with numeric features and multiple non-numeric classes (which are converted to numeric labels). Many variables can be tweaked such as number of decision trees, impurity algorithm, and maximum depth of the trees.

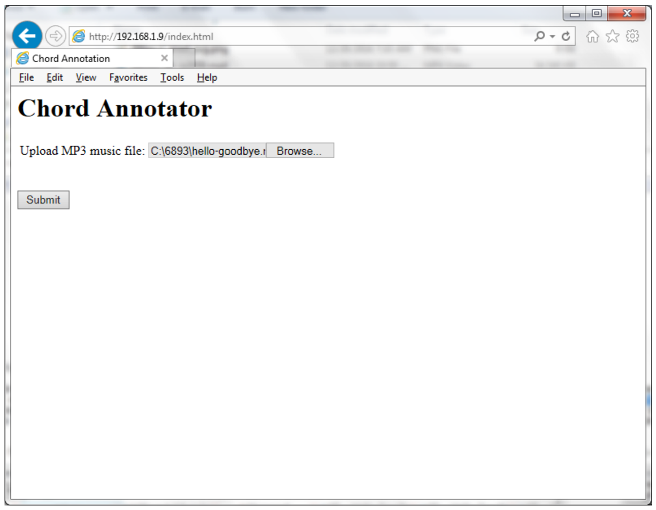
*Part iii*: The model can now be used in the backend of different types of applications which can extract chords from audio. I moved it to an Apache web server to be used on a web application. Therefore this Python program is a *Common Gateway Interface* (CGI) program which is a program that runs on the web server and generates dynamic web pages.

Once a user uploads an audio file, the Python program uses Librosa to once again separate the percussion from the harmony (HPSS) and generate chromagrams for each time frame. I divide up the song in 2 second segments. For each segment, a new record is created with a dummy chord and a feature value for each note based on the average value in the chromagram. Then the saved Spark model is used to predict the chord for each time frame, and it is printed out in HTML format and returned to the user.

## VI. Software Package Description

The application is a CGI web application. The Python program takes the uploaded audio file, extracts the features from each specified time frame, and runs each record through the saved random forest model using Spark. To run the application the Apache web server was started on my Mac with the command "*sudo apachectl start*."

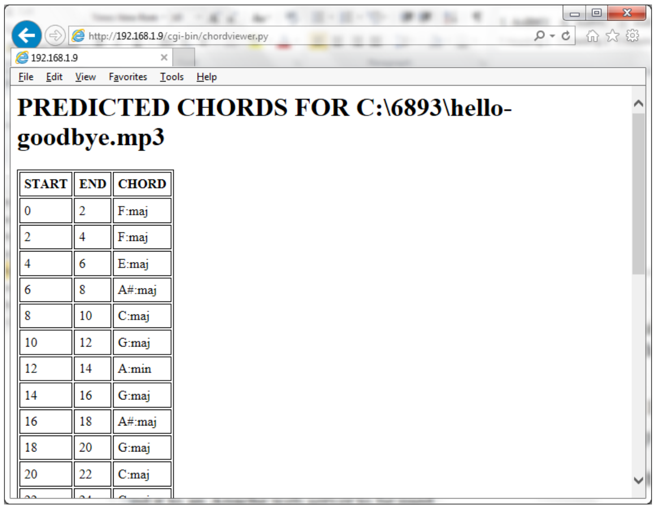
The user interface is an HTML file with a form to submit after uploading a song. Currently it works with mp3 format songs, although any format easily be configured. Here is the first screen as seen from a client computer:


http://192.168.1.9/index.html
Chord Annotation
File Edit View Favorites Tools Help
Browse...
Submit

Chord Annotator

Upload MP3 music file: C:\6893\hello-goodbye.

After the form is submitted to the server, the Python program predicts the chords and sends them to the client:


http://192.168.1.9/cgi-bin/chordviewer.py
192.168.1.9
File Edit View Favorites Tools Help

**PREDICTED CHORDS FOR C:\6893\hello-goodbye.mp3**

| START | END | CHORD |
|---|---|---|
| 0 | 2 | F:maj |
| 2 | 4 | F:maj |
| 4 | 6 | E:maj |
| 6 | 8 | A#:maj |
| 8 | 10 | C:maj |
| 10 | 12 | G:maj |
| 12 | 14 | A:min |
| 14 | 16 | G:maj |
| 16 | 18 | A#:maj |
| 18 | 20 | G:maj |
| 20 | 22 | C:maj |

## VII. Experiment Results

In experimentation, I tried other machine learning algorithms from Spark before choosing the random forest classifier. For each algorithm, I split the data into 70% for training and 30% for testing. The accuracy rate of the predictions is given by the pyspark.ml.evaluation library. I tried Naïve Bayes, but it doesn't work well with numeric and continuous features. I also tried Sparks's Multilayer Perceptron classification which is a Neural Network classifier and Spark's decision tree algorithm.

With the random forest classifier, I experimented with a number of variables such as number of trees, the maximum depth, the impurity methodology, and the maximum bins. Each of these variables affects the prediction accuracy, but not by too much. I got up to around 70% accuracy.

With Python's audio library Librosa, I experimented with many different sampling rates and also with two methods for extracting chromagrams which are the Short Time Fourier Transform and the Constant Q Transform [12]. I chose STFT because it seemed to be more robust at handling different kinds of music.

## VIII. Conclusion

In conclusion, chord recognition is a very useful application, and it is now more attainable due to audio libraries like Librosa and others like it. Even people who are not part of the MIR community can now use these libraries for chord extraction. The highest accuracy in recent history has been over 80% [2], but that should be able to increase. Librosa is capable of extracting many more features out of audio than I have had time or knowledge to do. Due to time constraints, I also did not find the original audio for every annotated song – only a subset. Accuracy could have been greatly increased with more audio. So there is plenty room for improvement – maybe even more accuracy than someone with an inborn ear for music!

## Acknowledgment

THE AUTHOR WOULD LIKE TO THANK COLUMBIA UNIVERSITY PROFESSOR CHING-YUNG LIN AND HEAD TA ERIC JOHNSON FOR THEIR INFORMATIVE AND CHALLENGING COURSE ON BIG DATA ANALYTICS WITHOUT WHICH THIS RESEARCH PAPER WOULD NOT HAVE BEEN WRITTEN. I COULDN'T HAVE ASKED FOR A BETTER COURSE TO COMPLETE MY DEGREE. THANK YOU TO COLUMBIA VIDEO NETWORK (CVN) WHICH ENABLED ME TO GET MY MASTER'S DEGREE IN COMPUTER SCIENCE AT A DISTINGUISHED UNIVERSITY WHILE WORKING FULL TIME. THE MACHINE LEARNING TRACK HAS BEEN FACINATING. THANK YOU TO MY PARENTS ISAAC AND LISA, AND MY GRANDMOTHER REGINA, AND NETFLIX FOR TAKING CARE OF THE KIDS WHILE I WAS KNEE-DEEP IN HOMEWORK OR STUDYING FOR GRUELING EXAMS. (NO, GRANDMA, YOUR NEIGHBOR WHO TEACHES QUICKBOOKS AND EXCEL WON'T BE ABLE TO HELP ME WITH THIS HOMEWORK.) THANKS TO HILLEL AND SHANA FOR ALL THE YUMMY MEALS. I WILL NOW BE ABLE TO MAKE YOU SOUP FROM SCRATCH. A BIG THANK YOU TO MY CHILDREN SARI AND SHUA FOR PUTTING UP WITH A MOTHER WHO IS GENERALLY FOUND IN FRONT OF HER COMPUTERS. I HOPE TO MAKE IT UP TO YOU. MOSTLY, THANK YOU G-D FOR EVERYTHING. IT'S BEEN A LONG RIDE AND I'M LOOKING FORWARD TO GETTING OFF THE COLLEGE TRAIN - DEGREE IN HAND.

## APPENDIX

Dataset links:

http://www.isophonics.net/datasets
http://ddmal.music.mcgill.ca/research/billboard

Converting youtube audio to different formats:

http://convert2mp3.net/en/

Github Respository:

https://github.com/Sapphirine/Machine_Learning_for_Chord_Recognition/

Youtube project link:

https://www.youtube.com/watch?v=BucCPu1yZrA

Big Data Analytics at Columbia:
https://www.ee.columbia.edu/~cylin/course/bigdata/

## REFERENCES

[1] https://www.boundless.com/users/232513/textbooks/understanding-basic-music-theory/the-physical-basis-3/acoustics-for-music-theory-18/wavelength-frequency-and-pitch-89-13565/

[2] http://www.mattmcvicar.com/wp-content/uploads/2014/02/thesis.pdf

[3] http://www.isophonics.net/

[4] http://ddmal.music.mcgill.ca/research/billboard

[5] http://convert2mp3.net/en/

[6] http://isophonics.net/content/reference-annotations

[7] https://en.wikipedia.org/wiki/Chroma_feature

[8] https://librosa.github.io/librosa/generated/librosa.core.stft.html

[9] https://ffmpeg.org/

[10] https://spark.apache.org/docs/1.6.0/api/python/pyspark.mllib.html

[11] http://stackoverflow.com/questions/31508083/difference-between-dataframe-and-rdd-in-spark

[12] https://librosa.github.io/librosa/generated/librosa.feature.chroma_cqt.html#librosa.feature.chroma_cqt