Talha Ansari
Reproducing Computation Research
May 16, 2014

Final Project Report
Data Mining Applied in Acoustic Bird Species Recognition

I. Introduction

In 2006, Dr. Anil Potti impressed the research community by his research on predicting the course of patients' lung cancer using microarray genetic analysis. His research garnered adulation from other's in the field and was considered a major breakthrough in the field of oncology. However, when other biostatisticians tried to verify the claims of the research, serious flaws were detected [1]. Eventually, it was concluded that the Dr. Potti's research was based on manipulated data. The resulting scandal, also referred to as the Duke research scandal, resulted in a major fallout in which various published papers and manuscripts were retracted. One wonders the impact scientific progress would have had if such malicious research had continued undetected. The reproducibility of research, which saved thousands of scientist hours and millions of taxpayers' money, is, therefore, of prime importance in maintaining the rigor, quality and efficiency of scientific research. Unfortunately, not many computational researchers give importance to the reproducibility of their work, making it difficult for others to verify their results. At Columbia University, the course Reproducing Computational Results is offered every year with the objective to encourage research practice which allows easy reproducibility. As part of the course, and to introduce students the problems of reproducibility, students are required to select a recent research paper and reproduce the results and figures of the paper as their final project.

This document is the write-up of my project in the class. I attempt to reproduce *Data Mining Applied to Acoustic Bird Species Recognition* authored by Erika Vilches, Ivan Escobar and Edgar

Vallejo from Monterrey Institute of Technology, Mexico and Charles Taylor from UCLA [2]. The selected paper explores the performance of three different machine learning techniques in classifying bird species using bird sounds. It also explores the effects of dimensionality reduction on accuracy of classifiers.

The rest of the report explains the project in more detail. Section II discusses the paper which I try to reproduce in great depth. Section III reports my reproduction efforts, focusing on issues that I face in replicating the paper's procedure, how I solve the hurdles, and how and why my experiments diverge from the original. Section IV presents the results and figures of my research and compares it with what is presented in the subject paper. Finally, Section V is the conclusion which summarizes the effort, draws conclusions from the results, and talks about the lessons learned from the project.

II. Discussing original paper

The paper which I attempt to reproduce explores the performance of machine learning techniques in classifying a bird specie using bird sounds. The motivation behind authors' research is a project going on in a Mexican ecological reserve, the aim of which is to install node like devices throughout the reserve to keep track of bird movements. Three bird species, namely great antshrike (Taraba major), dusky antbird (Ceromacra tyrannia) and barred antshrike (Thamnophillus Doliatus) are included in the research. These particular species are chosen because of their abundance in the respective ecological reserve.

The authors get their data from The Cornell Lab of Ornithology, Macaulay Library [3]. They use a total of 204 sound files, 49 for great antshrike, 79 for dusky antbird and 76 for barred antshrike. Each sound file is passed through a lowpass and a high pass filter, different for each specie, before being loaded into a software called Sound Ruler [4], an outdated software for sound analysis with no current technical support available. Sound Ruler allows authors to study the oscillogram and spectrograms of the sound files, and automatically locate calls and pulses

from the recording. Each recording consists of multiple calls, and each call consists of multiple pulses. A total of 21360 pulses for great antshrike, 5373 pulses for dusky antbird and 911 pulses for barred antshrike are generated. For each pulse, authors generate 71 features. These features are generated using Sound Ruler. Authors do not provide much information about the kinds of features which are generated. Thus, one of the crucial parts of the research is missing from the paper.

Authors explore three data mining techniques to test the validity of their idea, two decision tree algorithms ID3 and J4.8, and Naive-Bayes. All three classifiers are implemented using Weka Software [5]. As the two decision tree algorithms in Weka work only with discrete valued features, authors use a technique called vector quantization to convert numeric feature values into nominal. They use the ID3 and J4.8 decision tree algorithms to decide on the most important attributes of the feature set, reducing the total number features to 47, and then use Naive-Bayes on the reduced dataset. The results, with a cross-validation ratio of 70:30, show an accuracy of above 90%. Authors also claim that the performance on the reduced dataset is better than performance on the original dataset.

III. Reproduction

The reproduction part of *Data Mining Applied to Acoustic Bird Specie Recognition* is a challenging task. It can broadly divided into four broad groups of tasks. First, the data needs to be acquired, second, data needs to be pre-processed, third, feature values have to be extracted, fourth, classification algorithms need to be run. For each of these parts, I describe the challenges I face in replicating the same part from the paper, how I make assumptions when faced with lack of information, and how these assumptions possibly affect the final results.

1. Data acquisition

Just like the original paper, I use data from the Cornell Lab Ornithology, Macaulay Library. A total of 398 sound files are provided (400, with 2 corrupted) covering all the three species. The

data is provided in terms of individual downloadable sound files, making downloading a tedious task. As a result, I create a python script to do this automatically. However, the script is not the most efficient as it does not download the files directly but use internet browser to access the downloadable link. There is a discrepancy between the number of files I have and the number used in the original paper. Unfortunately, there is no way to find out the exact sound recordings which the original paper used. As a result, I shortlist about 228 recordings by looking at their size. Table 1 shows the breakdown of the the files by species and provides a comparison with the original paper.

Specie	No. of sound files in original paper	No. of sound files in reproduced
Taraba Major	49	68
Cercomacra tyrannia	79	79
Thamnophilus Doliatus	76	81
Total	204	228

Table 1. Comparison of number of files in the data, by species.

2. Sound Ruler

In the original paper, the entire data-preprocessing and feature extraction is done on Sound Ruler. Sound Ruler is an outdated software which is no longer under-development. It is designed to work on Windows, Apple Powerbooks, Linux and in Matlab. However, I find it really hard to set up Sound Ruler in either of these. Apple Powerbooks are no longer available. In Linux, it requires the installation of packages which were extremely difficult to find and also needs graphic functionality. For Matlab, Sound Ruler is only compatible with older versions does not work with the version available at Columbia University. Windows computer, on the other hand, was not easily accessible. Even when tried on Windows, SoundRuler, despite starting, crashed very frequently. After weeks of depending on Sound Ruler without getting any returns, I decided to

skip the software entirely and implement the necessary functionality myself, using a combination of Matlab and Python.

3. Segmentation into calls and pulses

The downloaded sound files were passed through filters, different for each species, as shown in Table 2. In place of Sound Ruler's automatic call and pulse detection, I write my own algorithm for detecting segments within a recording. I segment my original recordings by separating time-periods in which a sound is made. This is, unfortunately, more generic than Sound Ruler's pulses and more similar to Sound Ruler's calls. In order to locate pulses instead of calls, I try going more granular in terms of my parameters but the segmentation algorithm fails. A better way to find pulses, however, is an iterative approach - detect the calls using the segmentation algorithm and then run the segmentation algorithm on individual calls. Initially, this did not give good results as most of the call was still detected as a single pulse. Figure 1 below shows an oscillogram of a call which is further segmented into pulses using the old segmentation algorithm. As can be seen, the majority of the call (middle section) is still treated as a single pulse. Decreasing the frame-rate of the smoothing algorithm during segment detection led to single pulses being detected as two pulses, and some still not detected. In Figure 2, I have tried decreasing the frame-rate of the smoothing in the segmentation and, although there is improvement in detecting the pulses in the center, extra non-pulses have started being detected as pulses too.

	Taraba Major	Cercomacra Tyrannina	Thamnophilus Doliatus
Low-pass filter	3597 Hz	4200 Hz	3597 Hz
High-pass filter	517 Hz	920 Hz	686 Hz

Table 2. Filters used for pre-processing the sound files

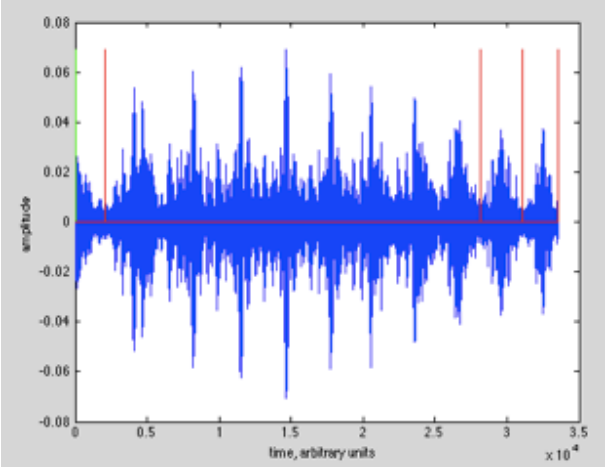

Figure 1. an oscillogram of a call which is further segmented into pulses using the old segmentation algorithm. As can be seen, the majority of the call (middle section) is still treated as a single pulse.

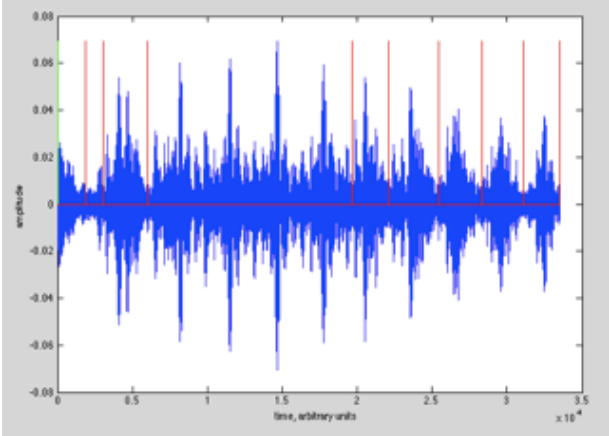

Figure 2. Old segmentation algorithm after decreasing the frame-rate of the smoothing in the segmentation. Although there is improvement in detecting the pulses in the center, extra non-pulses are being detected as pulses.

Finally, after much effort, the segmentation algorithm is improved upon and made to detect a good percentage of pulses without splitting them unnecessarily. The key change I make is in how each data point is compared to the data points around it, in terms of its amplitude and its 1st difference. The pseudo code for the algorithm is shown in Table 3. The improved results of

segmentation carried out on a randomly selected call are shown in Figure 3. Despite giving good results, the algorithm was extremely time-inefficient. Breaking each call of N elements into pulse required operations at an order of $O(N^2)$. To generate all the pulses from all calls (which had already been detected and stored) took 17 hours to finish on my Macbook Air.

```
Segmentation Algorithm (detecting pulses)

1. Given: audio array y, other parameters
2. Take the absolute of y, y => abs (y)
3. Smooth y (twice), y => smooth (y)
4. Take first difference of y, dy = diff (y)
5. declare var STATUS => OFF; pulse_start = []; pulse_stop = []
6. declare var win => ~1000
7. For each point in y:
      a. A = y(i)<mean(y(i:i+win))
      b. B = mean(y(i-win:i)) < mean(y(i-win:i+win))
      c. C = mean(y(i:i+win)) > mean(y(i-win:i+win))
      d. D = y(i)<mean(y(i-win:i))
      e. if (STATUS==OFF) & A & B & C:
         a. pulse starts
         b. STATUS = ON
      f. else if (STATUS==OFF) & D & B & C:
         a. pulse ends
         b. STATUS = OFF
8. Form pulses using the pulse start and end points
9. For each pulse:
      a. end = ending point of pulse
      b. start_next = starting point of next pulse
      c. If (start_next != end)
         a. end, start_next = mean (end, start_next)
```

TABLE 3. Pseudo-code of the segmentation algorithm used to detect pulses in calls

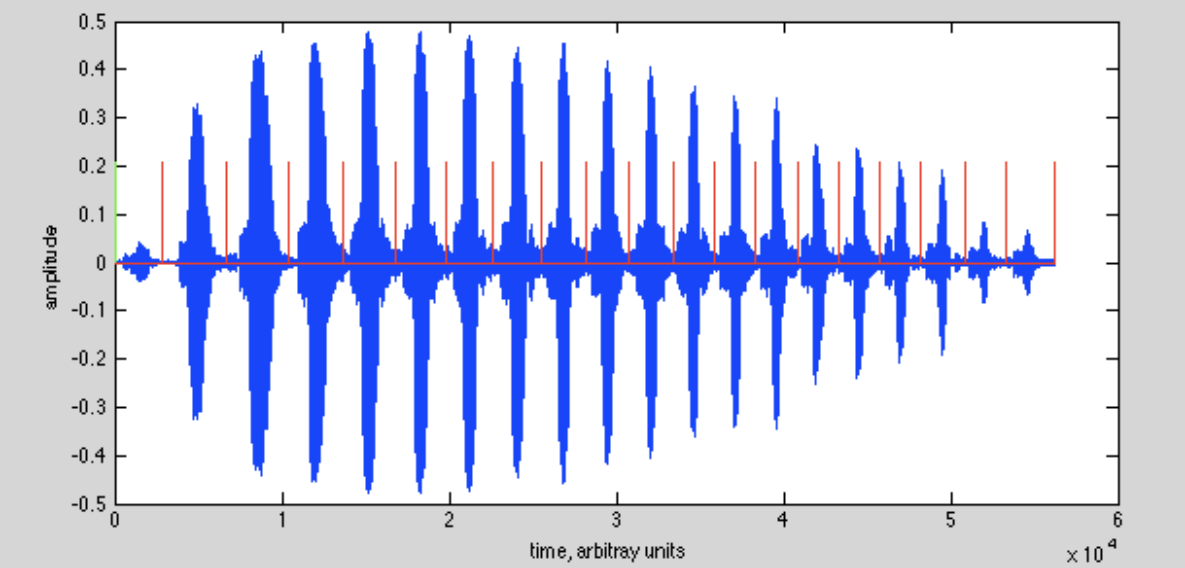

Figure 3. Oscillogram of a randomly chosen call, showing the effective pulse detection by the improved segmentation algorithm.

4. Maintaining the sample between species

From a total of 228 sound files, a total of 4960 calls and 58565 pulses are generated. The distribution of these pulses across species is, however, much different from what used in the original paper. The ratio used in the original paper is approximately 21:5:1. Therefore, in order to better match the original data, I remove randomly chosen pulses from two of the three species. Table 4 shows the number of pulses in the original dataset compared to my dataset, both pre and post the ratio fix. It needs to be noted that the ratio was fixed using the number of pulses. This is important because the final samples that I use in the dataset are pulses. Fixing the ratio either using the number of files or using the number of calls may not result in the same final ratio, as the number of pulses generated by each file or call is not the same.

Specie	no. of pulses in original	no. of pulses in reproduced	no. of pulses in reproduced after ratio fix
Taraba Major	21,360	20539	20539
Cercomacra tyrannia	5,373	20401	4890
Thamnophilus Doliatus	911	17625	880
Total	27644	58565	26309
Ratio	21:5:1	21:21:18	21:5:1

Table 4. Number of pulses in the original paper compared with those during reproduction, both before and after fixing the ratio.

5. Discussion on Feature selection

There was some discussion on using calls as the samples in the training data instead of pulses. The authors use a single feature vector for each *pulse,* one of the few things they are explicit about. Also, using calls instead of pulses as samples in my dataset leaves me with a much smaller sample set than what is used by the authors. Given that I feel confident about the performance of my segmentation algorithm, it makes sense to go ahead with pulses.

Although the paper does not mention most of the features which they use, five of the 71 features, namely Pulse dominant frequency, width dominant frequency, number of pulses, and dominant frequency at the final 50% of the call, and maximum pulse dominant frequency in a call, are mentioned in the results section of the paper. Pulse dominant frequency is the frequency of the highest peak in the FFT of the entire pulse. Width dominant frequency is a measure of the bandwidth of the highest peak, i.e. the bandwidth at -10dB, as a proportion of the center frequency. The samples in the research were individual pulses (sub-segments of calls),

so all the pulses which come under the same call have the same number of pulses, and the same for dominant frequency at the final 50% of the call and the maximum dominant frequency in a call. Out of these four mentioned features, I was not able to effectively calculate the width dominant frequency, so the feature was not included in my training data.

This, however, leaves me with another problem. The authors report the use of 71 features and I only have 4 out of them. With not much information regarding other features, I decide to use Mel-frequency coefficients (MFCC). MFCC are widely used in the domain of sound recognition and have shown compelling results. Voice Box, a signal processing toolbox for Matlab, built by Mike Brooke [5], is used to generate the 12th order MFC coefficients. The features are the summary statistics of the MFC coefficients, such as the mean, min, maximum, variance, mean of 1st differential etc. The final dataset have 59 features, 55 of which are MFFC features and the rest four are the pulse dominant frequency, number of pulses, dominant frequency at the last half of the call and the maximum pulse-dominant frequency in a call.

6. Vector Quantization

The paper implements the two decision tree algorithms, ID3 and J4.8, and Naive-Bayes. However, the decision tree algorithms accept only discrete feature values. Therefore, as explained by authors, I carry out vector quantization - a process to convert continuous feature values into discrete, given a set number of bins. Vector quantization basically divides the continuous range of values into a set number of bins and then assigns each value to the bin corresponding to the range the value falls in. The pseudo algorithm for vector quantization in shown in Table 5 and the resulting performance is compared with the original paper in Figure 4.

```
Segmentation Algorithm (detecting pulses)

# FUNCTION: Vector Quantization - Carries out Vector Quantization
# RETURNS: Feature data with numeric values replaced by nominal values

1. Given a feature vector F, and number of bins
2. maxF => maximum value of F
3. minF => minimum value of F
4. step => (maxF - minF)/bins
5. partition => vector initialize with length bins
6. codebook => vector initialize with length bins + 1
7. for j in range(bins+1): # initialize the partition vector and codebook
      a. partition[j] = (minF+(step*j)
      b. codebook[j] = j
8. nominalF = vq(column,partition) # vq is from Python's scipy.cluster
9. return nominalF
```

Table 5. Pseudo-code of vector quantization algorithm

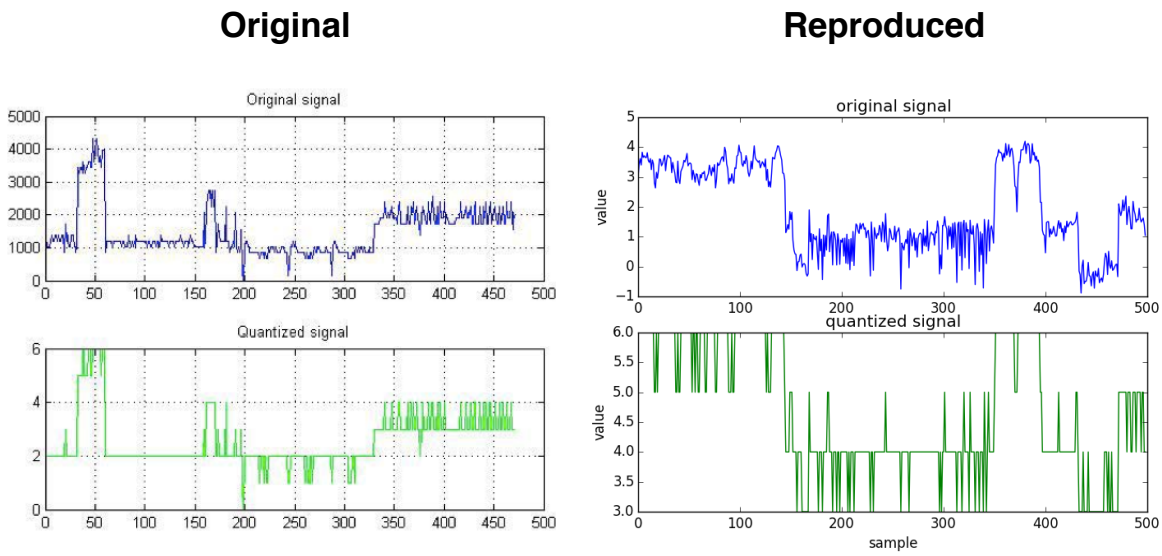

Figure 4. Comparison of vector quantization results between the original paper and the reproduction. Similar results can be seen.

7. Classification

Most of the classification part of the project was replicated without any discrepancies. Weka Software [6] is used to implement the three classifiers. Unlike Sound Ruler, Weka Software is supported by a strong developer community and is therefore easily implementable. Weka provides a lot of functionality in the domain of machine learning, including many data processing filters, classifiers, and post processing analytical tools. It also carries out cross-validation. For this research, a cross-validation with a training-testing split of 70-30% was selected.

As suggested by the original paper, I tried five approaches towards classification. First and second, I test the accuracy of the decision tree algorithms ID3 and J4.8 on the entire feature set. Third, I run Naive-Bayes on the entire dataset. Fourth, I reduce the dataset by including only the features selected by ID3 in the first approach, and then run Naive-Bayes on the reduced the dataset. Fifth, I reduce the dataset using features selected by J4.8 and then run Naive-Bayes on the reduced dataset. The different approaches to classification are depicted in Figure 5. The results of these classifiers are shown and compared with the original results in Section IV.

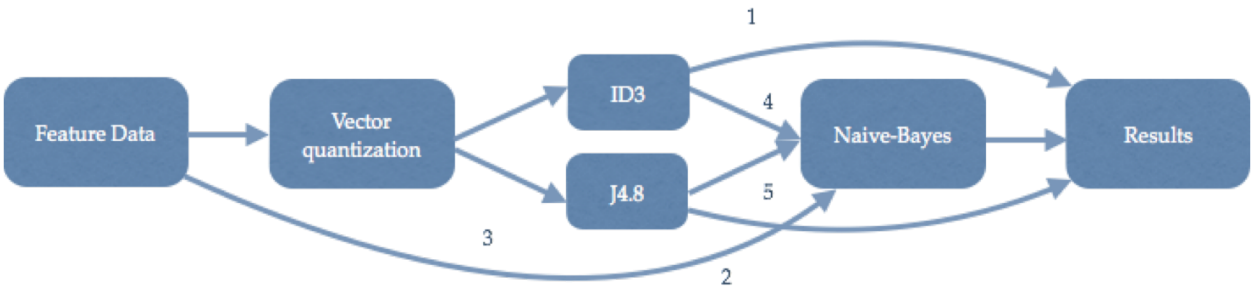

Figure 5. Five different approaches to classification implemented by the original paper and this reproduced version, the results of which are compared later.

8. Technology structure and Github Reproduction Package

All of the code used in my reproduction project has been put on Github repository [7]. For this project, I use a combination of Matlab, Python and Weka Software. The audio file reading,

pre-screening of audio files, segmentation into calls and pulses and generation of MFCC coefficients are carried out in Matlab, while the conversion of MFCC coefficients into features, extraction of four frequency features, vector quantization and correction of species distribution ratio are carried out in Python. Once the feature data is created, Weka is used to run the classifiers. The output from the ID3 and J4.8 classifiers is again used to reduced the dataset, which is then again used in Weka to test the performance of Naive-Bayes on the reduced datasets.

IV. Results and Comparison

Out of the available 59 features, the decision tree algorithms ID3 and J4.8 choose 45 and 38 features respectively. In contrast, the feature set was reduced to 47 from 71 in the original paper. The authors do not mention which exact decision tree algorithm leads to this feature drop but, given authors' focus on J4.8, I think it can be assumed that it is due to J4.8.

The features chosen by decision trees in my work are mainly MFFC features. Interestingly, the frequency features occur lower down the tree. The root, in both cases, is one of the mean of the MFCC. In contrast, the authors report that the J4.8 had the most important feature (root node) of dominant frequency followed by the width dominant frequency, the maximum pulse dominant frequency in a call and dominant frequency in the last 50% of the call. The relative importance given by the decision algorithms to the frequency features is, therefore, different in our result. This is perhaps due to the inclusion of the more powerful MFCC features which possibly dominate over the frequency features.

Accuracy results of each of the five classification approaches are obtained directly from the Weka software and are reported in Figure 6 below. The authors claim, that the use a J4.8-reduced dataset on Naive-Bayes leads to a 4.5% increase in accuracy, is also not supported by my results. In contrast, the reduced dataset leads to a decrease in accuracy by 1 percent point. As can be seen, the accuracies achieved in my experiment for each of the five approaches are slightly

better than those reported in the original paper. This, again, can be due to the presence of the MFCC features in our dataset. However, just as reported in the original paper, J4.8 is found to be the most accurate of the classifiers. In Figure 7, the accuracy of J4.8 with respect to specific species is shown and compared to with the original paper. Although the accuracy of J4.8 in case of Taraba major and Cercomacra tyrannia is close to what was claimed by the authors, the accuracy for Thamnopilus doltatus is found to be about 12% lower than what was reported. Overall, it can be seen the results do, in general, follow the same theme as those claimed by the paper.

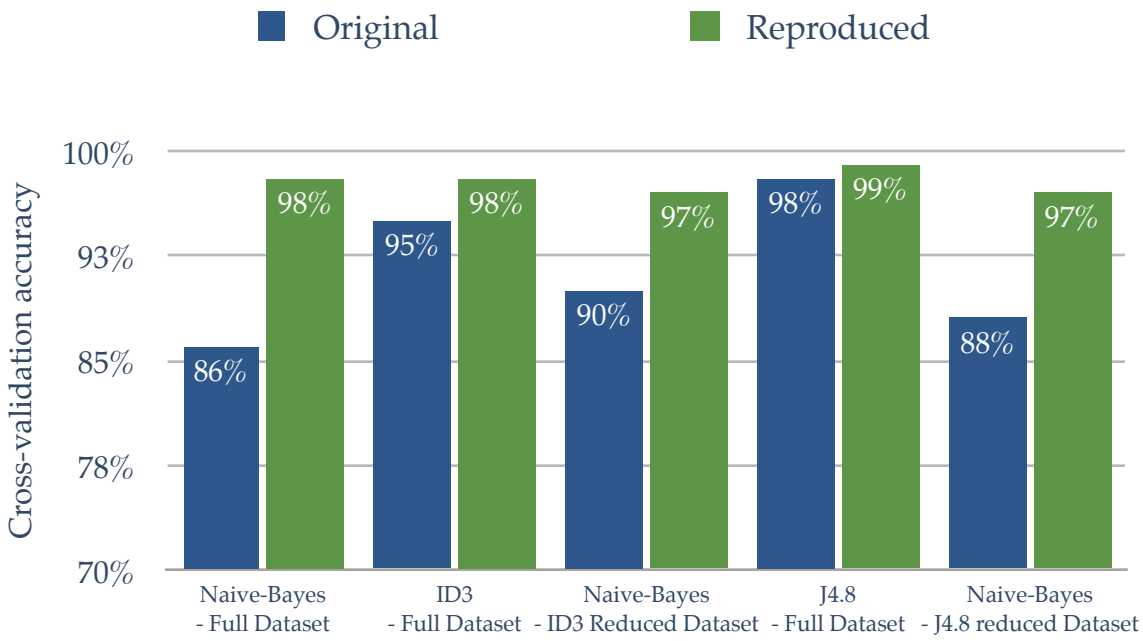

Figure 6. Correct classification rates of five different classification approaches, compared between the original paper and what I obtained.

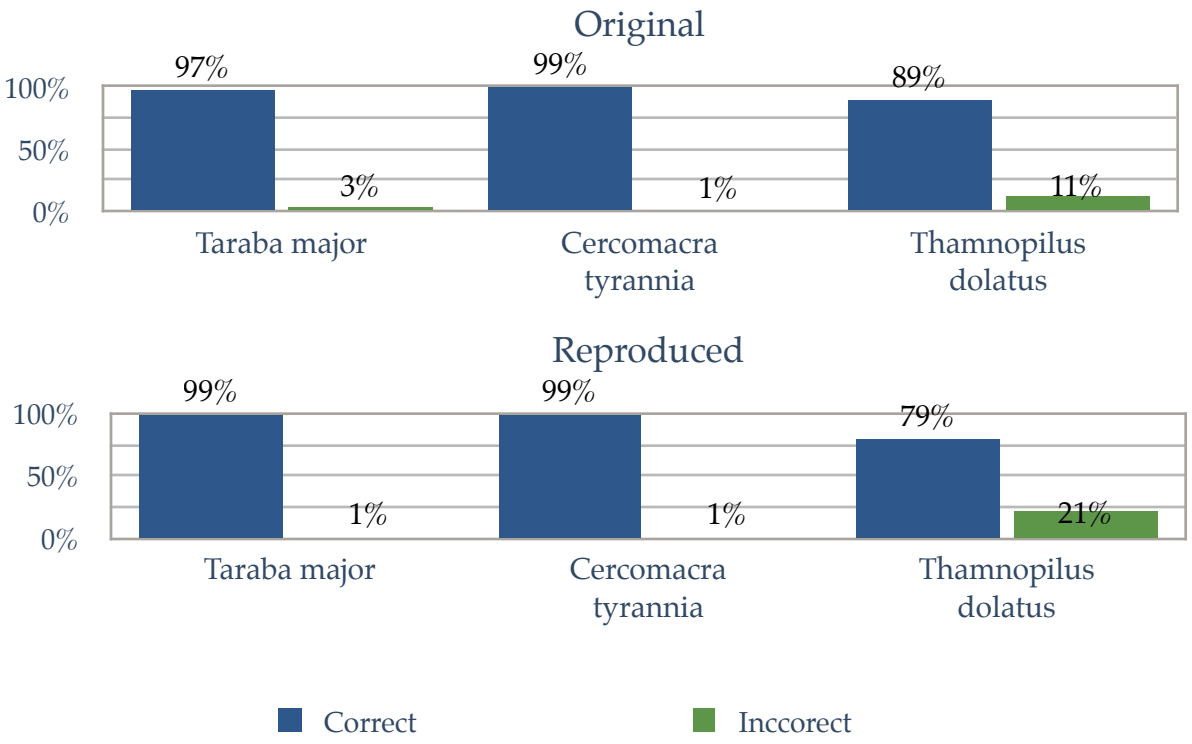

Figure 7. J4.8 Accuracy percentage graph compared between the original paper and what I obtained.

V. Conclusions

The reproduction of *Data Mining Applied to Acoustic Bird Specie Recognition* was, as expected, not hurdle free. Whether the findings of the paper, Data Mining Applied to Acoustic Bird Specie Recognition, were verified or not is a subjective question. Although the results I obtain do match those obtained in the original paper, the unfortunate digressions in my method make it difficult to make any strong conclusions.

The biggest single issue in the project was the lack of compatibility of Sound Ruler. Although the authors do not have any control over the future availability of the third party tools that they use, the problems setting up Sound Ruler did make the reproduction much harder. A lot of effort was not only spent in trying to get the software to work, but also in replicating some of the software's functionality.

Several factors influence the quality of the reproduction.There are chances that sound recordings that I use differ somewhat from those used in the original paper. One can also wonder how closely my code for segment detection, despite its promising results, matches that of Sound Ruler. Finally, my features, which were heavily based on MFCC were most probably not the exact features which the authors use. But, I think I had no choice. Removing them would have left me with only a handful of features as compared to authors' 71 features. Given the strength of MFCC features, one can expect the frequency features to be easily dominated, and therefore not contribute as much during the classification process, which is what I think happened.

Another important aspect of the paper worth discussion in the distribution of the data across species. In the final dataset, the number of pulses are distributed with a ratio 21:5:1, which is extremely skewed. Biased distribution of the samples, which, if not accounted for, leads to biased results. When Weka Software was run with the option of making the samples per species equal in the training set, the classification accuracies dropped to below 30%. Given that this option is turned off by default and that the authors don't mention it in the paper, it is assumed that the authors run their classifiers without this option. This, therefore, raises serious questions over the conclusions that the authors draw from their results

Scientific research, with the advent of computers and their ever increasing computational power, depends highly on computer simulations and data-mining techniques today. Unlike in the past, where reproducing a research experiment required expensive apparatus setup, computational results can be reproduced almost exactly if provided with enough information and the code, allowing researchers to verify each others' work and, hopefully, discourage scandals like the Duke research scandal.

VI. References

[1] Kolata, Gina (7 July 2011). "How Bright Promise in Cancer Testing Fell Apart". New York Times. Retrieved 17 January 2012.

[2] Vilches, Erika, et al. "Data mining applied to acoustic bird species recognition." Pattern Recognition, 2006. ICPR 2006. 18th International Conference on. Vol. 3. IEEE, 2006.

[3] Cornell Lab of Ornithology - Macaulay Library, http://www.birds.cornell.edu/macaulaylibrary

[4] Sound Ruler, http://soundruler.sourceforge.net, by Marcos Gridi

[5] Mike Brooke. Voicebox: Speech Processing Toolbox for Matlab, http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html

[6] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[7] Ansari, Talha. Columbia_e689, http://github.com/talhajansari/columbia_e6891.git