Task 1:

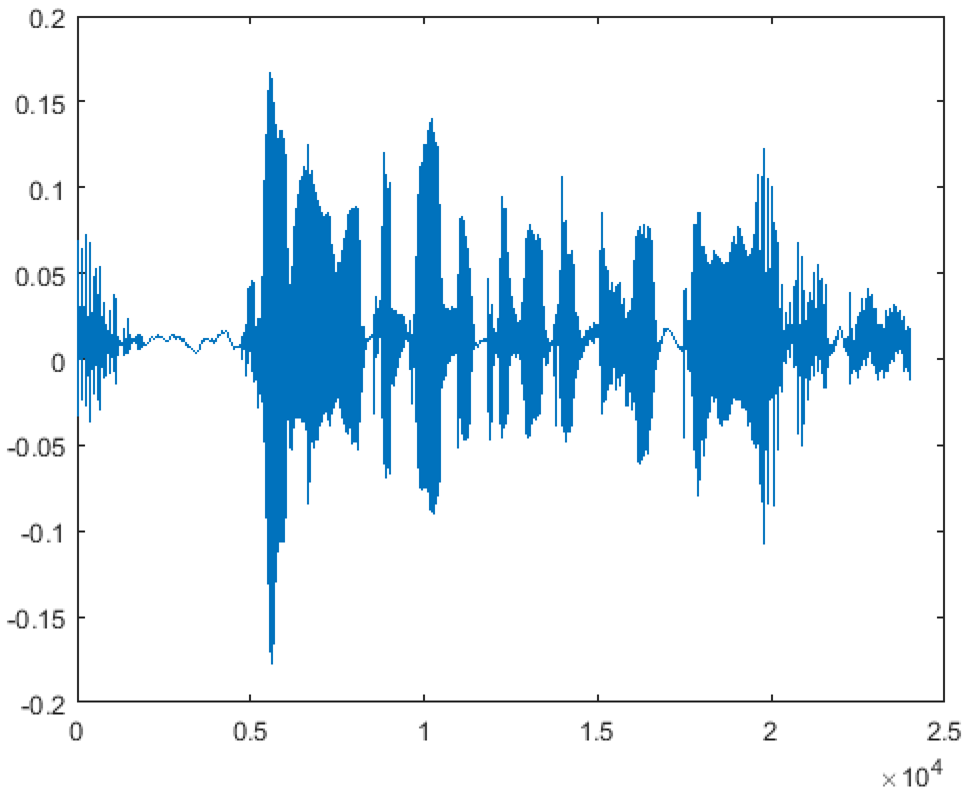
I obtained the first two rows of data from the matrix trainAudiorecords by using the following two commands.

```
row1 = trainAudiorecords(1,:);

row2 = trainAudiorecords(2,:);
```

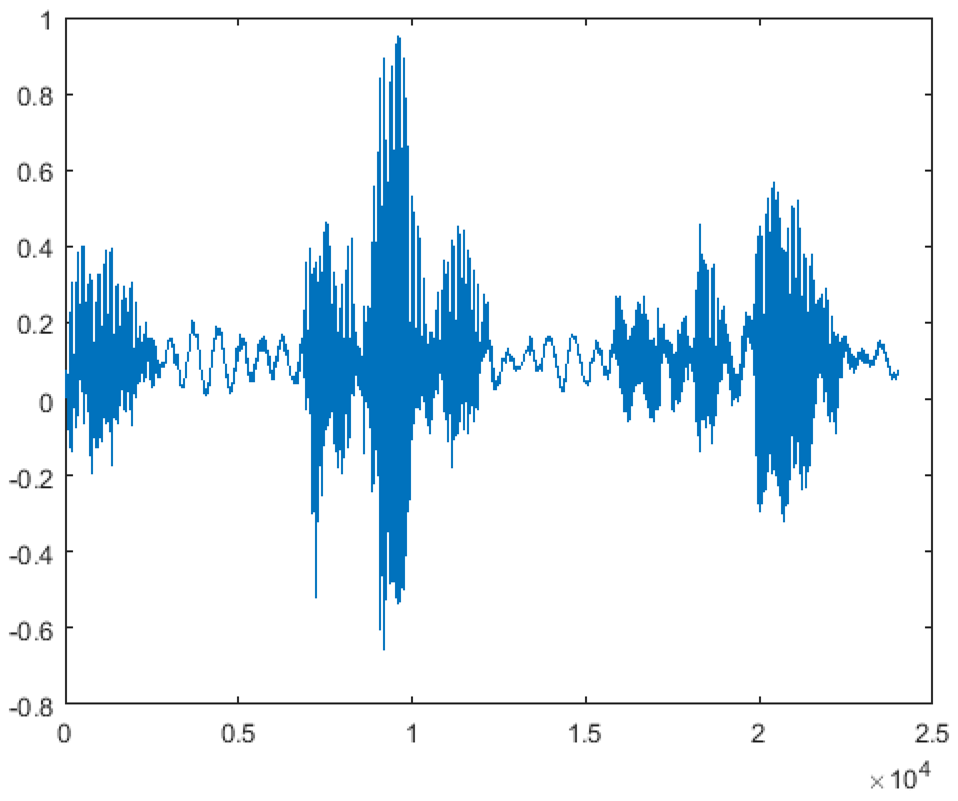
I then plotted the signals to verify that I properly grabbed two separate signals.

```
Plot(row1);

Plot(row2);
```

Graph of Row1



Graph of Row2



The resulting audio signals are distinct with row1 corresponding to a 1 and row2 corresponding with a 0 according to the values of the matrix trainAudiolabels. Once I obtained this information I created the corresponding wav files using writeaudio. The following two function calls are what I used to create two audio files voice1.wav and voice2.wav.

```
audiowrite('voice1.wav',row1, 8000); audiowrite('voice2.wav',row2, 8000);
```

Voice1.wav is a 3 second clip of a woman saying, “tell me what to do because he doesn’t have priority.” Voice2.wav is a 3 second clip of a male saying, “and I would love that and she said.” Since voice 1 corresponds to a 1 and voice 2 corresponds to a 0 in the training data, it can be concluded that 1 corresponds to a female voice and 0 corresponds to a male voice.

Task 2:

I ran the following to train the svm classifier.

trained = svmtrain(trainAudiorecords, trainAudiolabels);

It outputted the following warning, but executed successfully.

Warning: svmtrain will be removed in a future release. Use fitcsvm instead.

Using the classifier I ran the following to see how well it predicted the voices in the testdata.

result = svmclassify(trained, testAudiorecords);

It executed successfully, but printed out a similar warning as the one for svmtrain. To compare how well it did when compared to the vector testAudiolabels, I wrote the following matlab function that takes two vectors as inputs and returns what percent of the vectors were the same.

```
function [m] = VecCmp(V1, V2)

size = length(V1);
size2 = length(V2);

if(size ~= size2)
   print("Vectors must be of same length");
   m = -1;
   return;
end

count = 0;

for i = 1:size
    if(V1(i) == V2(i))
        count = count + 1;
    end
end

m = count / size;
end
```

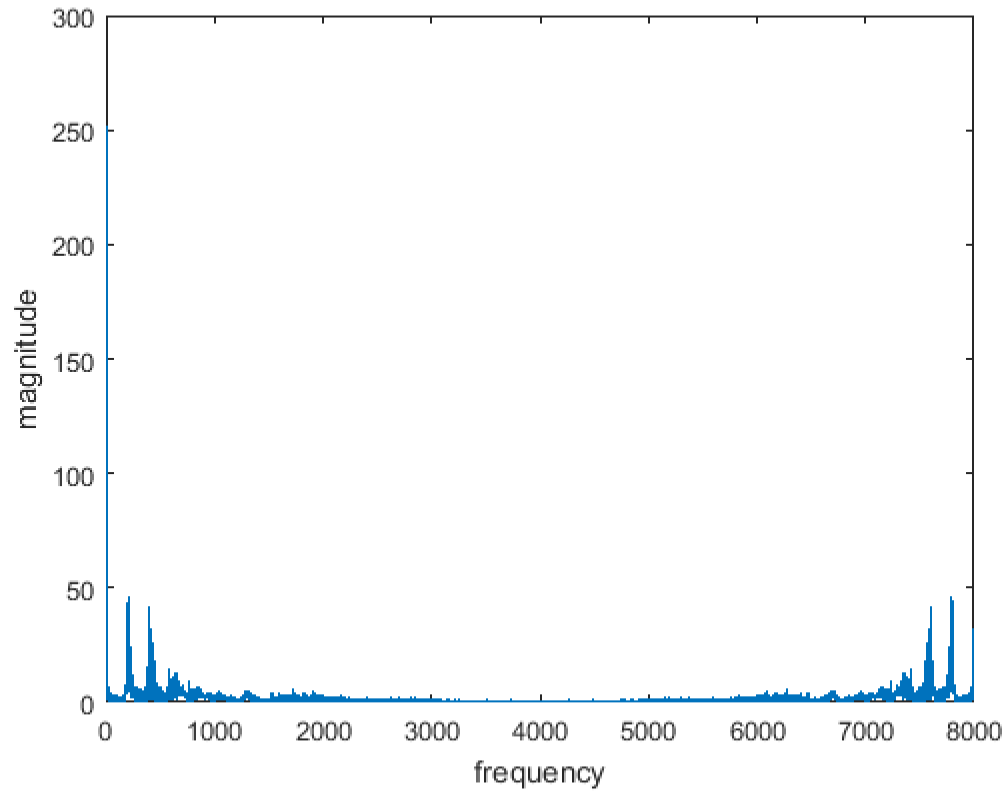
When I ran VecCmp with testAudiolabels and result as inputs I got that the trained svm was accurate for 57.3% of audio signals.

Task 3:

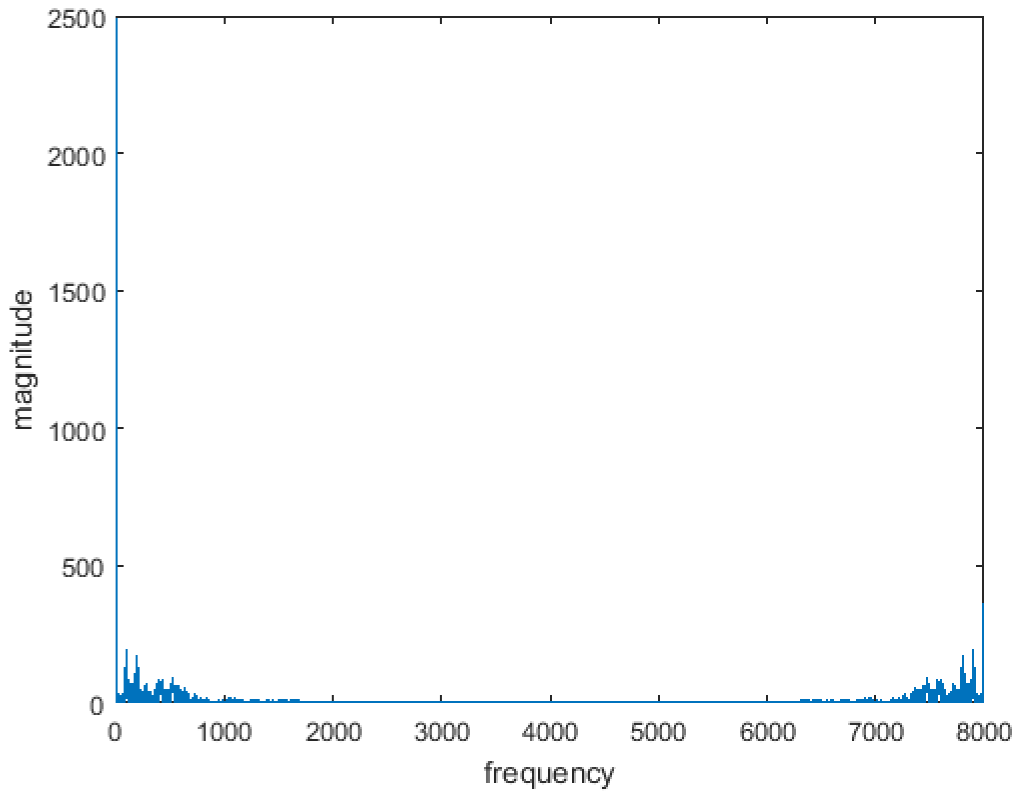
The vectors row1 and row2 contain the data for rows 1 and 2 of the Matrix trainAudiorecords. The code to create the fourier transform and plot the magnitude of these first two rows is as follows.

```
freqrow1 = fft(row1);

freqrow2 = fft(row2);

magnitude1 = abs(freqrow1);

magnitude2 = abs(freqrow2);

faxis1 = linspace(0, 8000, 24000);

plot(faxis1, magnitude1)

 xlabel('frequency');

 ylabel('magnitude');

plot(faxis1, magnitude1)

 xlabel('frequency');

 ylabel('magnitude');
```

Magnitude1 Graph



Magnitude 2 Graph



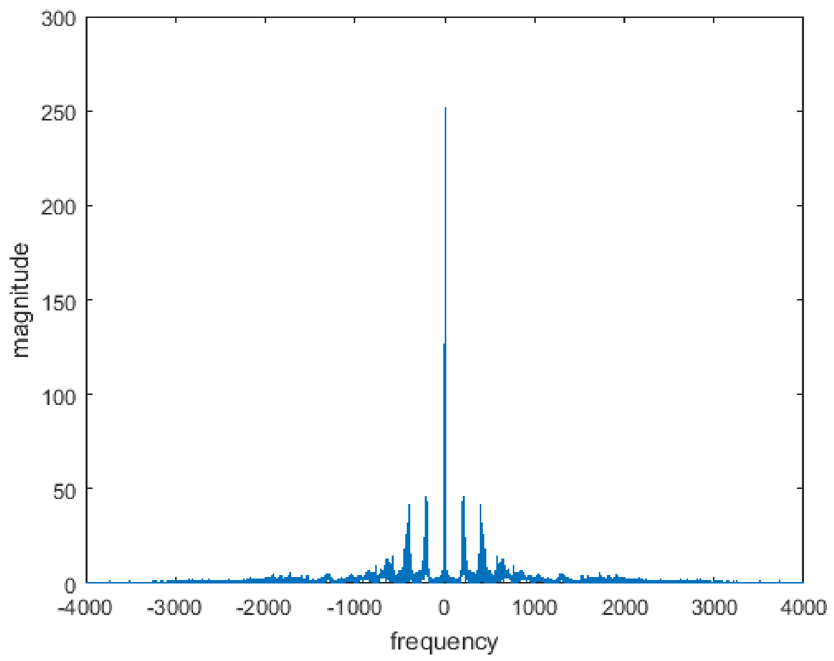
The first graph has much lower frequency magnitudes than the second graph. Both graphs tend to have peaks at similar locations on the frequency axis. Also another major difference is the frequency ranges on the first and second graphs. The first graphs displays a longer range of noticeable frequencies than the second.

The code used to create the zero centered versions of these signals is as follows.

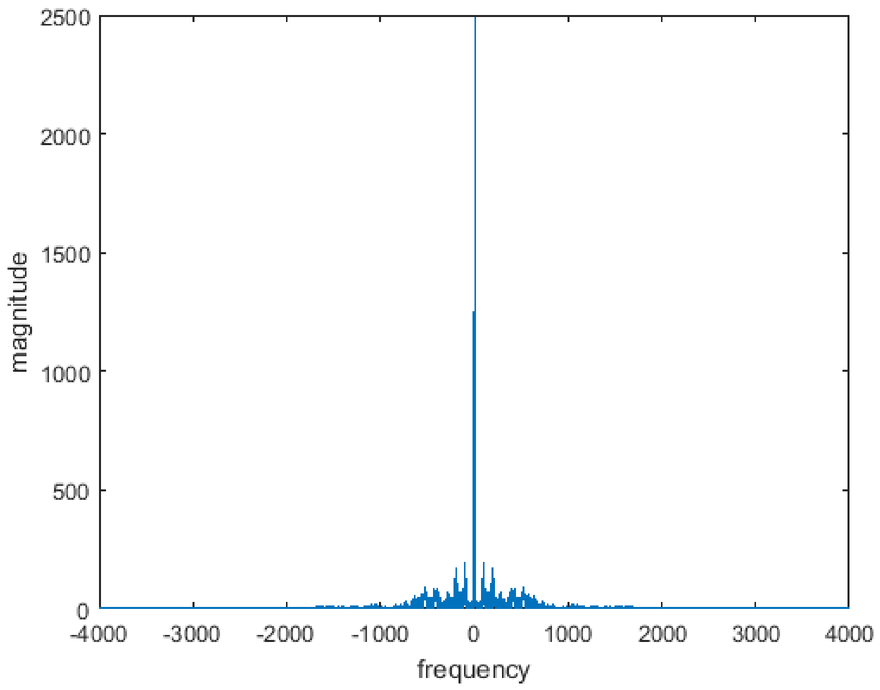
```
shift1 = fftshift(freqrow1);

shift2 = fftshift(freqrow2);

shiftmag1 = abs(shift1);

shiftmag2 = abs(shift2);

plot(faxis2, shiftmag1)

xlabel('frequency');

ylabel('magnitude');

plot(faxis1, shiftmag1)
```

```
xlabel('frequency');

ylabel('magnitude');
```

Shifted Magnitude 1 Graph



Shifted Magnitude 2 Graph



Task 4: I wrote the following function to calculate the power spectrum of a matrix where the fft should be applied rowWise rather than columnwise.

```
function [m] = PowerSpectrumRowWise(M)

transposed = transpose(M);
freqTransposed = fft(transposed);
magnitude = abs(freqTransposed);
powerTransposedaudiorecords = magnitude.^2;
m = transpose(powerTransposedaudiorecords);
end
```

To obtain training and testing data as power spectrums I called PowerSpectrumRowWise with both train Audiorecords and testAudiorecords. To train the svm and obtain the results of the test I did the following.

```
powertrain = PowerSpectrumRowWise(trainAudiorecords);

powerTest = PowerSpectrumRowWise(testAudiorecords);

powertrained = svmtrain(powertrain, trainAudiolabels);

powertested = svmclassify(powertrained, powertest);

VecCmp(powertested, trainAudiolabels);
```

The results were that it was capable of predicting the correct speaker 96.89% of the time. This is a major improvement over the original trial.

Task 5:

I attempted to train the svm by passing in a vector of the powerbandwidth of the audio signals. After doing some research on different aspects of male and female audio signals, I found that one of the distinctions between these audio signals is the length of the power band of the audio signal. I decided to use this to train my svm classifier and see how well it performs. I used the matlab function powerbw to generate an estimate of the 3dB bandwith of each of the signals and used them to train my svm. Upon completion it was successful in 63.5% of signal guesses. While not as great as the results from task 4, it did manage to outperform the method of task 2. The following is the function I wrote to calculate the bandwidth of the training matrix and the test matrix.

```
function [m] = bandLength(M, N)

test = zeros(N,1);
for i = 1:N
   test(i) = powerbw(M(i,:), 8000);
end

m = test;
end
```

I used a sampling rate of 8000 for the parameter to the function as the signals were sampled at 8000 Hz. N is the parameter I used to specify the number of rows to make my matrix.