**Project 4 – Birdsong**

**Introduction**

The purpose of this project is to use Digital Signal Processing and Machine Learning to understand the movement, growth, behavior, and migration patterns of regional birds around the Georgia Tech Campus. For this project, noise induced due to internal and external sources affects the ability to design and build an optimal classifier for bird noises. Hence external noise such as environmental noise, white noise, human conversations, automobile noise must be removed from the sample files in order to only operate on bird sounds. Of these noise sources, we chose to create a classifier that can help eliminate automobile noise from the existing audio sample recordings.

**Background**

In order to build these classifiers, we were provided with 625 audio samples collected from four locations on the Georgia Tech campus. Each of the 625 audio samples was one minute long, unique, and recorded at different times of the day. To begin exploring these files, we first carried out experiments to hear, and visualize the different audio components of these files. Next, we explored feature extraction techniques which could be used to extract specific characteristics from these audio files and set as inputs to train the support vector machines-based classifier. All programming and visualization tasks were carried out in MATLAB.

**Team**

**Angad Daryani** – Experiments, Feature Extraction, Report **Lillian Anderson** – Train and test with SVM, File Output

**Experiments**

For the purpose of this project, all the audio files were renamed from their 8-character alpha-numeric file names to 4-character filenames in order to simplify file handling while programming. For example, file name *‘5E6BA3C8’* was renamed as *‘ASB (1)’*. Locations were renamed as follows:

*Anderson-serviceberry-selected 🡪 ASB   
ArchitectureWest-selected 🡪 AW  
CherryEmerson-selected 🡪 CE  
EBB-selected 🡪 EBB*

Step 1: Hearing the audio file and picking out characteristic sounds pertaining to automobiles

Being Georgia Tech students, we knew the physical areas where these audio files were recorded at, and hence it was easy to understand that there would minimal automobile audio noise in the files pertaining to the Architecture West Building. Hearing all 20 audio files validated this information since the background noise was either white noise, environmental noise, or from construction occurring in front of the architecture building. On hearing more audio samples from the different locations, we were able to identify 6 file samples with audible automobile noise, along with their corresponding times of occurrence in the one-minute clip.

|  |  |
| --- | --- |
| **File Name** | **Times of occurrence** |
| ASB (1) | 0.30 to 0.41 |
| ASB (2) | 0.27 to 0.29 |
| ASB (17) | 0.10 to 0.14 |
| CE (1) | 0.23 to 0.27 |
| CE (24) | 0.14 to 0.19 |
| EBB (1) | 0.14 to 0.27, 0.39 to 0.48 |

**Table 1. Audible automobile noise in sample files with corresponding times**

Step 2: Visualizing the audio file components pertaining to automobile sounds

We used Audio sample EBB (1) and from hearing the file, we could identify that the first moving automobile sound occurred around 22s, and hence on clipping the audio file to be between 20 and 24s, and by performing an FFT on it, we were able to identify the frequency range within which the moving automobile sound existed to be 20.25kHz to 27.75kHz.

A screenshot of a map

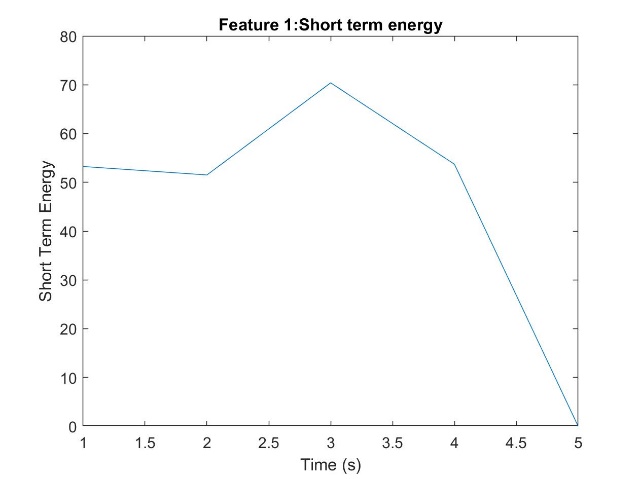
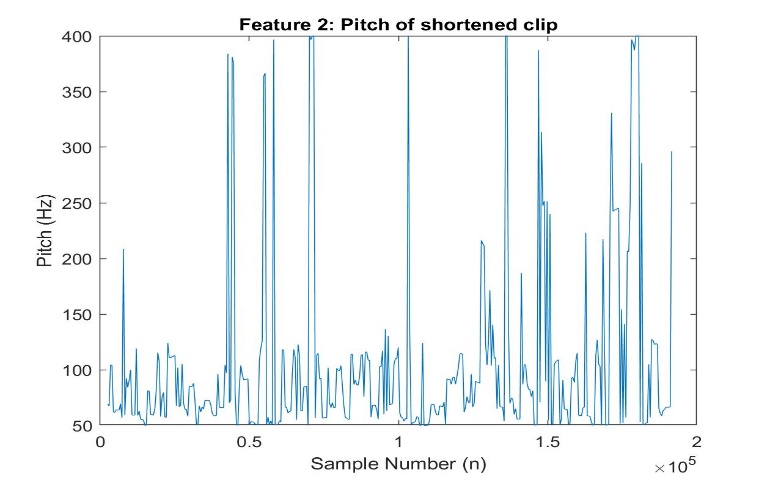
Description automatically generated

Automobile Noise

**Figure 1. FFT of shortened audio clip for EB (1)**

**Feature Extraction**

In order to be able to consistently identify the appropriate components of the audio files which could be compared to a training dataset to further classify the noise source, it was important to explore several feature extraction techniques. For the purpose of this project, we explored the feature extraction techniques of MFCC, short-term energy, pitch, wavelet packet transform, short time Fourier transform, spectral flux, and spectral centroid. Of these, we chose to only use short term energy and pitch after our testing of all these feature extractors with SVM. The following examples are demonstrated on the shortened audio clip for ‘EB (1)’.

White noise

STE Peaks at 22s

**Feature 1: Short Term Energy Feature 2: Pitch**

A picture containing screenshot

Description automatically generated

Automobile Noise

**Feature 3: Wavelet Packet Transform**

A close up of text on a white surface

Description automatically generated  
**Feature 4: Time Domain Spectral Flux**

Each of the features displayed above highlight unique aspects of the shortened audio clip. ***Short term energy*** helps classify the voiced, unvoiced, and silence regions of an audio clip. While there is no unvoiced region in this clip, we can see a strong distinction in the energy when the car is closest to the recording device at 22s. The ***Pitch*** is the fundamental period of a signal, the perceptual correlate of fundamental frequency. From our experimenting, we noted that car engine pitch was consistently in the range of 75 – 250 Hz, while the bird song was higher pitched. This matches with what we hear in everyday life. The ***Wavelet Packet Transform*** is a mathematical means for signals with varying frequencies and can sometimes provide better analysis of a signal than other feature extraction techniques. Lastly, the ***Spectral Flux*** measures the rate of change of spectral content with time. It is computed by taking the square of the differences between consecutive spectral samples. Each of these unique features serves as an important tool for the automated classification of environmental sounds, and in our case background automotive sounds.

In selecting a useful feature, it is also important to find distinct features between the classifications. To do so, we compared the various feature extraction methods side-by-side when applied to car sounds and bird sounds. **Figure 2** exhibits how the car’s time domain spectral flux has a distinct peak around 2.3 seconds. The bird’s flux is fairly constant across time, but it does have a lull at 2.3 seconds. We can predict from this that the car samples would be labeled accurately with this feature extraction. If anything, there would be a few false positive car labels.

A picture containing clock

Description automatically generated

**Figure 2. Car and Bird Comparison of Time Domain Spectral Flux**

**Classification**

Classification is the computational method to automatically sort different noise sources in our case, into distinct buckets of information. Using this, we are better able to remove these noise sources and attain the most important information which is bird sounds. There are several automated methods of classification such as Support Vector Machines, Convolutional Neural Networks, Regression, Trees, and Nearest Neighbors. Since we had limited amount of training data as well as fewer dimensions, we chose to use Support Vector Machines (SVM). SVM helps maximize the margins for separable data. In order to analyze our results with SVM we carried out the following steps:

1. Prepare the dataset, dividing it into training and testing sets
2. Feature extraction for training and testing (we looked at 4 features and used the best 2)
3. Train the support vector machine model and plotting the hyperplane
4. Apply the trained SVM model to test set and analyze the likelihood scores

We have detailed descriptions of these steps mentioned below along with our algorithm performance.

Step 1. Prepare Dataset

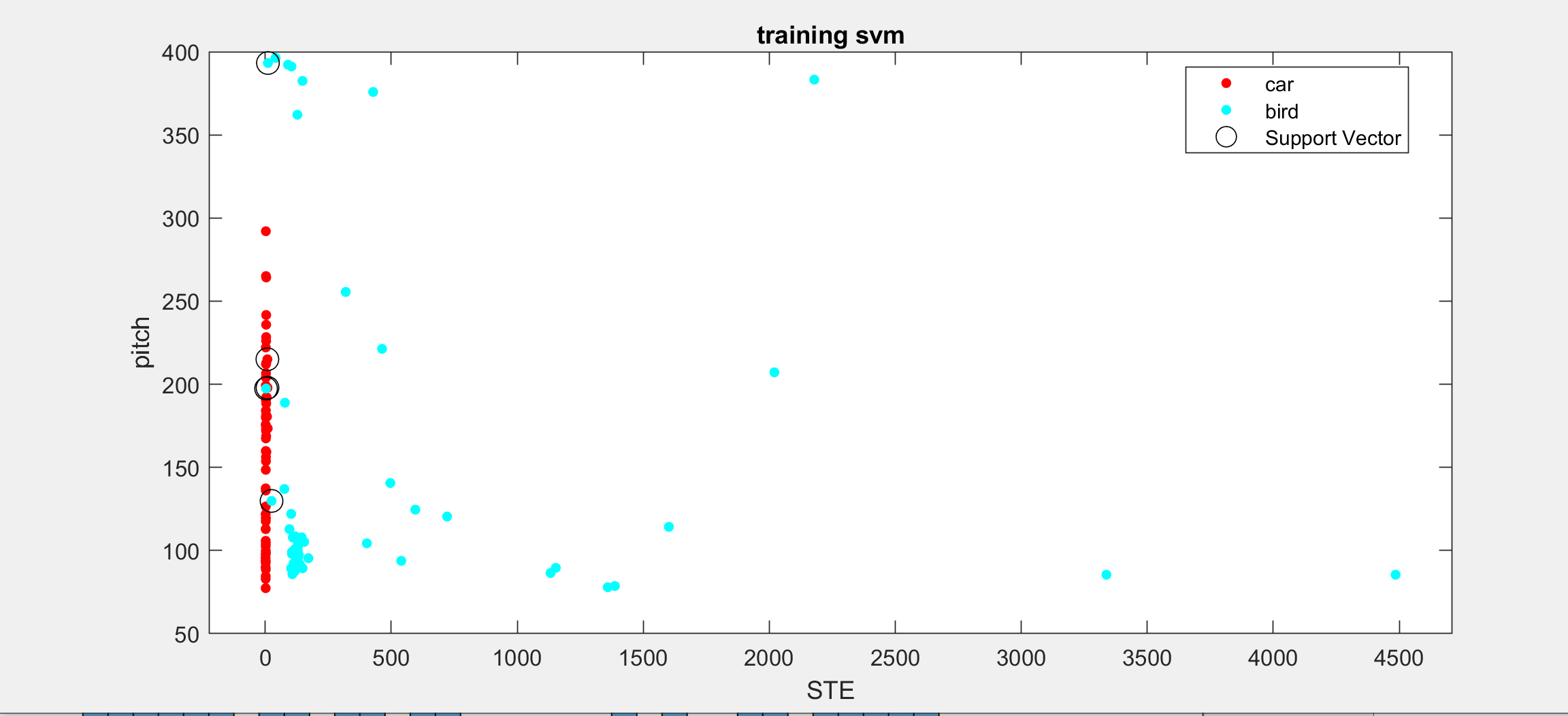
We created a .wav file specifically for training. The file contained 1 minute of various car drive-by sounds found on the internet concatenated with 1 minute of ‘ASB (29).wav’ which contains various bird sounds and no automobile noise. The file has 3,840,000 samples to train from. This set was easily labeled “automobile” for the first half of the samples and “bird” for the second half. The testing sets were the provided audio files for this project.

Step 2. Feature Extraction

We tested various combinations of the feature extractions discussed above and decided to use pitch and short-term energy as the two main features because they had the most distinct SVM training results (see step 3). Feature extraction was applied to both the training and testing files.

Step 3. Train the Support Vector Machine Model

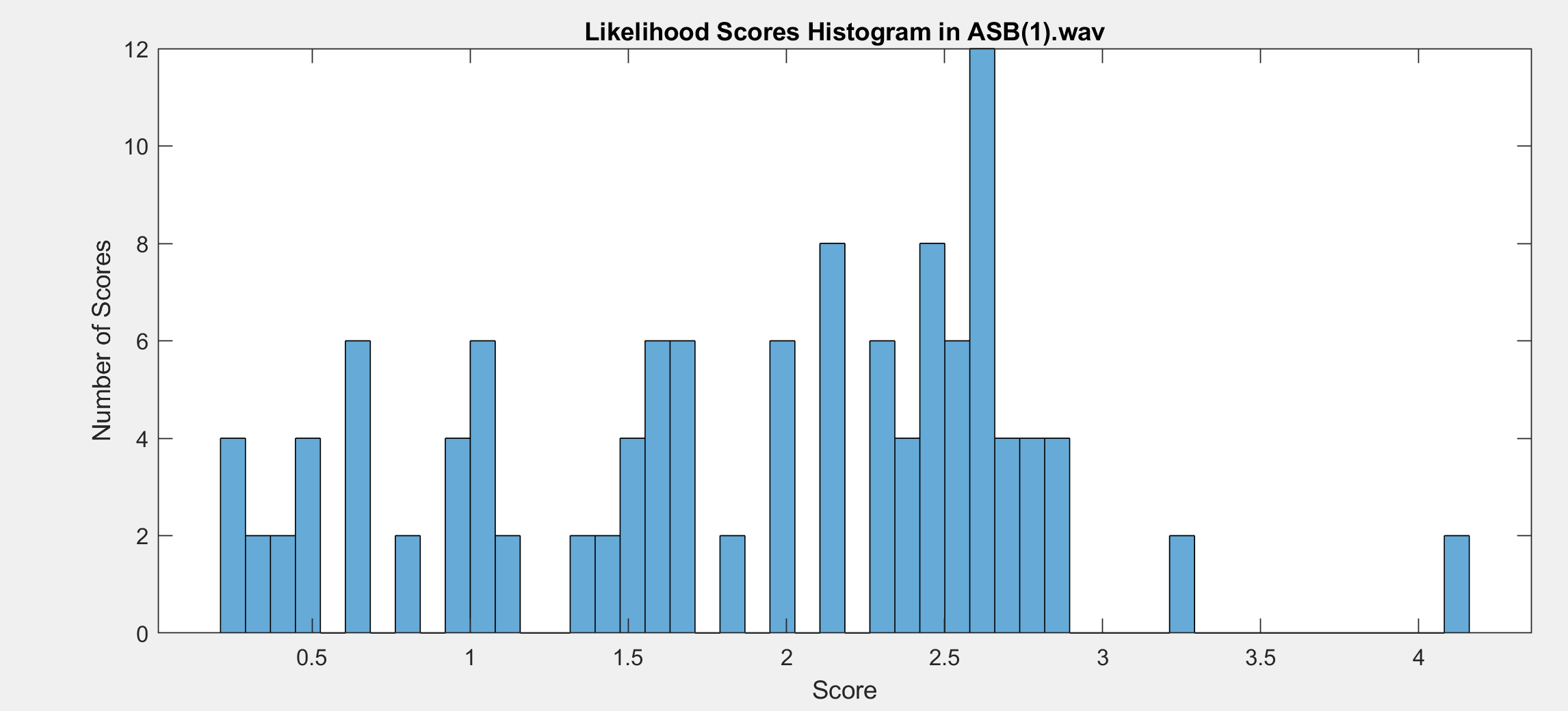
An SVM works by plotting feature points with known classes and then drawing a hyperplane to separate the classes. To train our SVM, we fed our training .wav file and labels into the fitcsvm() MATLAB function and plotted the results, as seen in **Figure 3**. The support vectors, circled, help determine the hyperplane tilt and location for maximizing the margin. For our SVM model, there is a strong distinction between automobile and other noise. The bird class contains several different bird chirps, quiet moments, and non-automobile noise, so the points are much more scattered for that group.



**Figure 3. SVM Training Results**

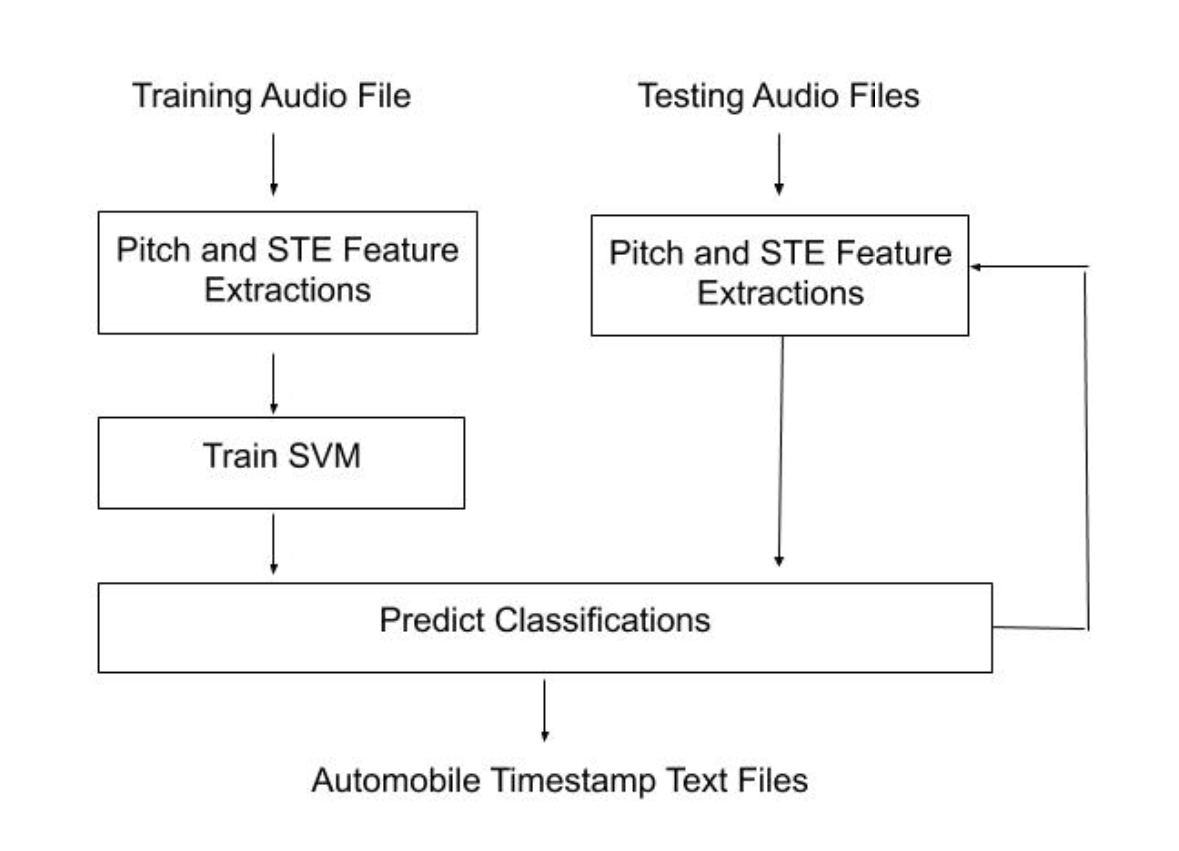
Step 4. Apply SVM to Test Sets

After an SVM model is found, we can use it to predict classifications for the test sets using the MATLAB function predict(). The prediction function plots the features for a sample and notes where it is in relation to the learned hyperplane. The histogram below shows the likelihood scores that the classifications assigned are accurate. Notice that no sample has a score of 0, which would show uncertainty of the classification. Many of the samples have a likeliness score of 1.5 – 3, which is a reliable score.



**Figure 4. Likelihood Scores Histogram for ‘ASB (1).wav’**

A flowchart of the code structure is displayed in **Figure 5**. All of the code is implemented in a single script in MATLAB. The feature extractors are implemented as subfunctions. After the SVM has been trained, the code loops through each testing audio file and reads it in, extracts features, predicts classifications, and then outputs results in a text file.



**Performance**

We tested the feature extractor and classifier with 13 unique audio files (ASB 1 to 10, and ASB 13,19,24) and found that for most of the cases, the performance was near perfect. The time at which the audio was heard was very close to or included the timestamp that the algorithm output. However, in a file like ASB 13, it completely missed the automobile noise in the first 10 seconds and misinterpret the combination of bird noises and water droplets as automobiles at 24s and 29s. Similarly, in ASB (19), it missed the automobile sounds from 13s to 21s. For most of the other audio files, the occurrence of automobile noises falls well within the timestamps issued by the algorithm. However, there were false positive reporting of automobiles from background noise such as someone tinkering in a garage, and the wind. All these sounds also have lower pitch and similar energy content, so our algorithm wasn’t accurate enough to make these differentiations. We found an average positive detection rate of 65% with this system.

**Conclusion**

SVM served as a good starting point for detection and removal of automobile noise. Better training data and use of more features for each test would help significantly improve the quality of detection. More advanced systems must be developed for automobile horns, electric vehicles, sirens, and doppler shift.