Urban Sound Classification Interim Report

Menna El-Shaer

May 2020

1. Problem Statement

The question we aim to investigate here is given a brief sound excerpt from an urban environment, what is the source of the sound? The goal is to develop a simple classifier system that can output the sound label after learning an appropriate data classification model from a set of training samples that have been manually annotated. System performance will be measured using a classification accuracy score using an independently labeled test set of sound samples. The developed solution is of interest to audio processing system software devices. Classifying ubiquitous sounds according to their sources can be a preprocessing step for many audio processing systems before more sophisticated data analyses. For example, speech recognition devices need to remove non-speech sounds before language translation. As a result, identifying these non-speech patterns can help improve the recognition capabilities of many current speech recognition software.

2. Dataset

The dataset consists of 8732 sound excerpts that are shorter than 4 seconds each. Sound excerpts can be categorized into ten classes that represent common sounds often experienced in an urban environment. The labeled classes available in the dataset are grouped into the following sounds: Air Conditioner; Car Horn; Children Playing; Dog Bark; Drilling; Engine Idling; Gun Shot; Jackhammer; Siren; Street Music. The dataset is available for download at: https://drive.google.com/drive/folders/0By0bAi7hOBAFUHVXd1JCN3MwTEU. The training sample dataset is 3.4GB and the test sample dataset is 2.15GB. Data files consist of .wav audio file formats where each file has a unique numeric ID. The class labels are in a separate excel file in the same folder that has two columns containing the file ID and its class annotation. The training data is completely labeled while

the test data is not. This could be problematic in the accuracy score calculations part of the project since there is no ground truth. We can overcome this in two ways either by adding an extra labeling step to fill in the missing labels or using part of the training data files in the testing stage. It is to be noted, the same file cannot be used for training and testing simultaneously. Hence, we will use part of the training dataset in the testing stage. The test dataset files will not be used in this project.

3. Methods

A rough outline of the project stages is shown in Figure 1. We will use Python 3 packages to do all the data analysis, modeling and visualization needed to draw the appropriate conclusions.

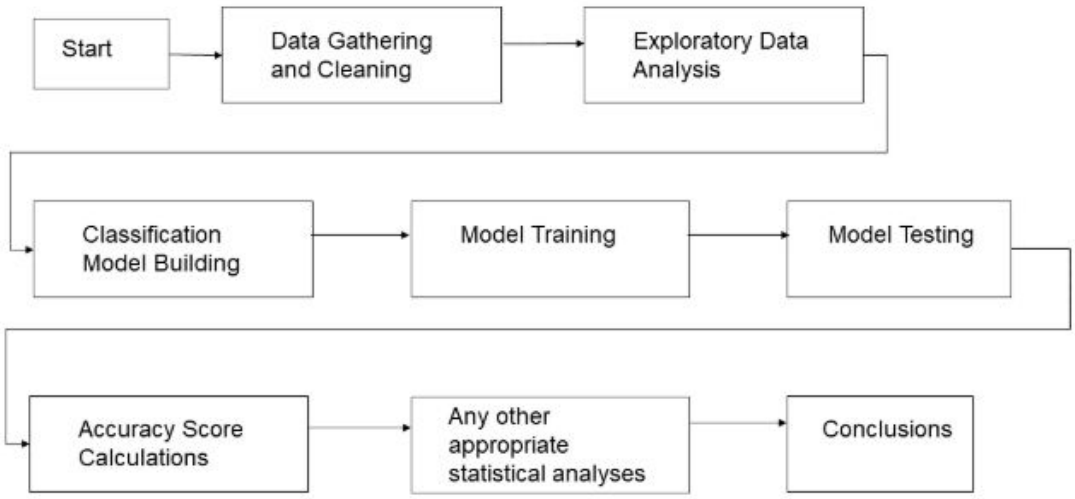

Figure 1: Project Steps and Milestones

3.1. Reading in the Data

After acquiring the data, the next step is to read in the data files and store them in a processing friendly format. The data is .wav audio file format, each data sample has one (mono) or two (stereo) channels. It is important to read in the properties of the audio files for data analysis and cleaning. Audio file properties extracted were sampling rate in Hz, sampling width in bytes, total number of frames and number of audio channels. These properties were then stored in a dataprop.csv for future reference.

3.2. Loading Data Frames

Python's wave module is the standard for loading .wav files. I utilized this module to extract audio properties in the previous section. A more numpy friendly approach to reading the data would be to use the scipy.io package [https://docs.scipy.org/doc/scipy/reference/generated/scipy.io.wavfile.read.html]. For example, for a 16-bit Pulse Code Modulated audio, reading the file produces a numpy array of int16 datatype in the range from -32768 to 32767. This numpy array can then be made into a pandas dataframe for statistical analysis. A drawback of using the scipy.io package would be its inability to read 24-bit sampled audio data. This is quite a popular format and thus some files in our data have a 24-bit sample width. Due to this issue, the functionality of reading 24-bit data is needed in the implementation. A separate issue is the inability of the wave module to read oversampled audio, in this case files with sample frequency greater than 48kHz. Thus, all oversampled audio will have to be resampled, this will reduce the data volume available for analysis later. We used ffmpeg [https://ffmpeg.org/] for data resampling. While resampling data, we made sure all output files have the same bit depth (16-bit PCM), this should help in the data cleaning stage.
Classification labels corresponding to training samples are provided in .csv format that could also be read in as a single numpy array. Alternatively, we will use the pandas dataframe structure and methods to handle the label data.

3.3. Data Cleaning

After reading and loading the labels data into a single dataframe, we need to set the index and columns appropriately as well as their data types.
The labels dataframe variable will have two columns (ID (dtype: int64); Class (dtype: object)). We will need to change the dtype of ID column to string or pandas "object" to match the ID column of the audio dictionary that has keys as ID stored as strings and values as int16 numpy array data. Just as we used the ID to index the audio dictionary, we'll set the index of the labels dataframe to ID.
Audio data is stored as a numpy array type in a python dictionary. Each element in the array column is the value of the discrete sound signal and multichannel values are in multidimensional arrays. In our case, the maximum number of channels is 2 (stereo), so at most an entry may be a 2D array.
There were no missing or null values in the labels dataframe or the audio dictionary.

3.4. Exploratory Data Analysis

3.4.1. Data Storytelling

After the data wrangling step, all the data are stored in a python dictionary with keys as file IDs and values as the discrete sound channel signal. In our dataset, we have 5435 sound files categorized into ten different types or classes. In this chapter, we will choose a sample size of one file for each class to explore and visualize the sound signal within. Randomly chosen sample files from each class were 915 (dog_bark),
1873 (jackhammer), 5678 (air_conditioner), 5289 (street_music), 6729 (drilling), 522 (children_playing), 7046 (siren), 6668 (car_horn), 2099 (engine_idling), 4171 (gun_shot).
Using spectrograms and wave plots, we will visualize and explore data features that could help in classifying sound signals. We'll start with signal amplitudes followed by signal power or energy analysis. We will explore some feature extraction techniques of audio signal data. The purpose of this step is to explore the data, generate interesting features and insights that should help in the classification part of the project. Extracting features requires us to operate on all files, however in this section we will visualize a subsample of the feature extraction results from the different classes. We will visualize, hypothesize and summarize the plots after.

Part 1: Wave plots and Spectrograms:

A wave plot is defined as a plot of the amplitude of the sound signal versus time. A spectrogram, however, is a plot of the signal's frequencies versus time. Independent axes represent time and frequency while color represents the signal's amplitude of the observed frequency at a certain time. The higher the color intensity, the stronger frequency it is present in the signal. This could be a very important feature in classifying different types of sounds. To plot a spectrogram, we need to compute the Discrete Fourier Transform of the signal to find the frequencies. Since our signal is a time series, a window frame of a pre-specified size is used for the Fourier Transform. In our experiments, we started with the default window size of 93ms. The resulting spectrogram 2D matrix has frames as rows, frequencies as columns and matrix values are amplitudes in decibels.

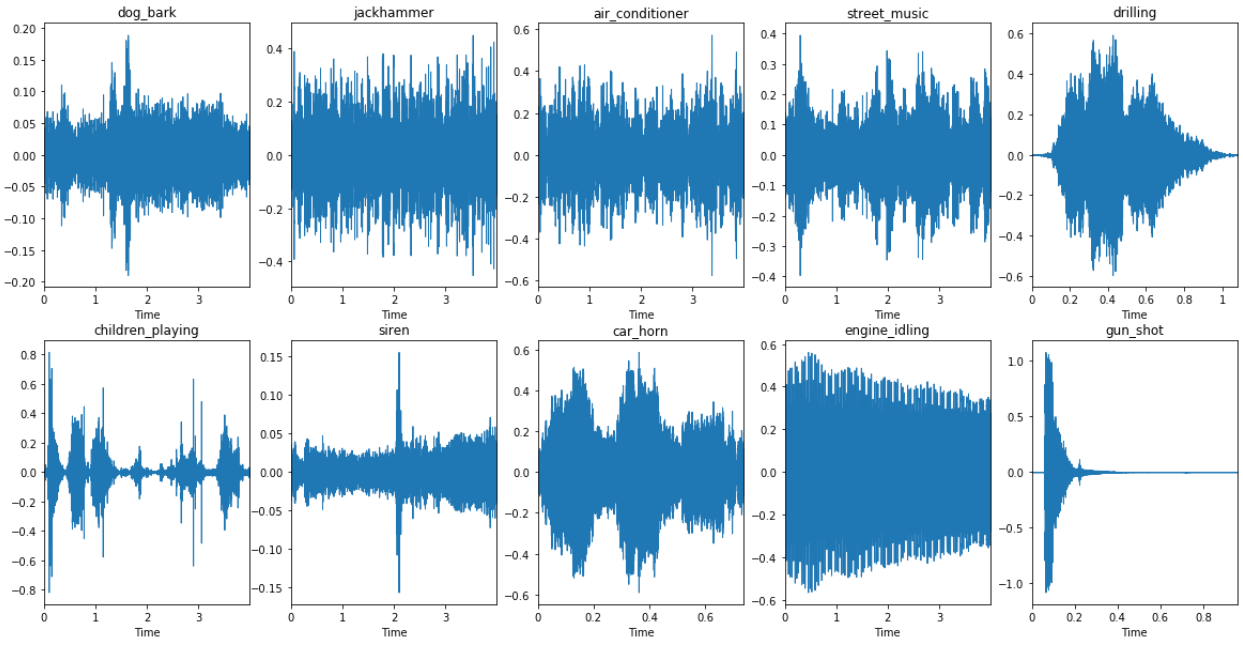

Figure 2: Wave plots of chosen samples

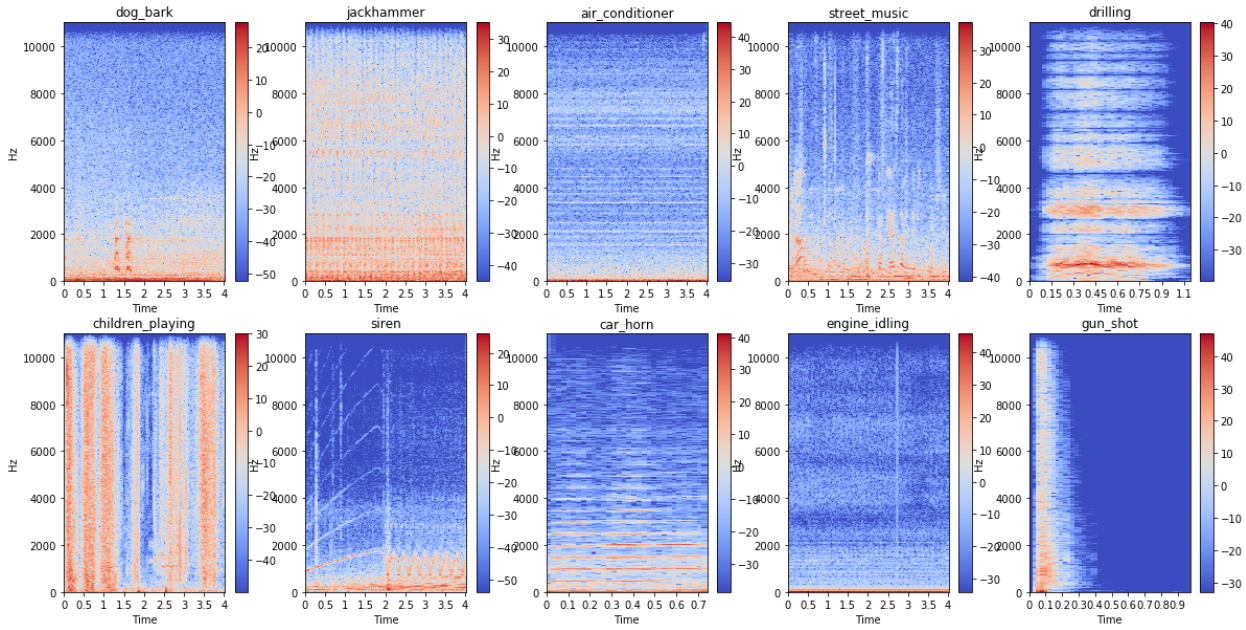

Figure 3: Spectrograms of chosen samples

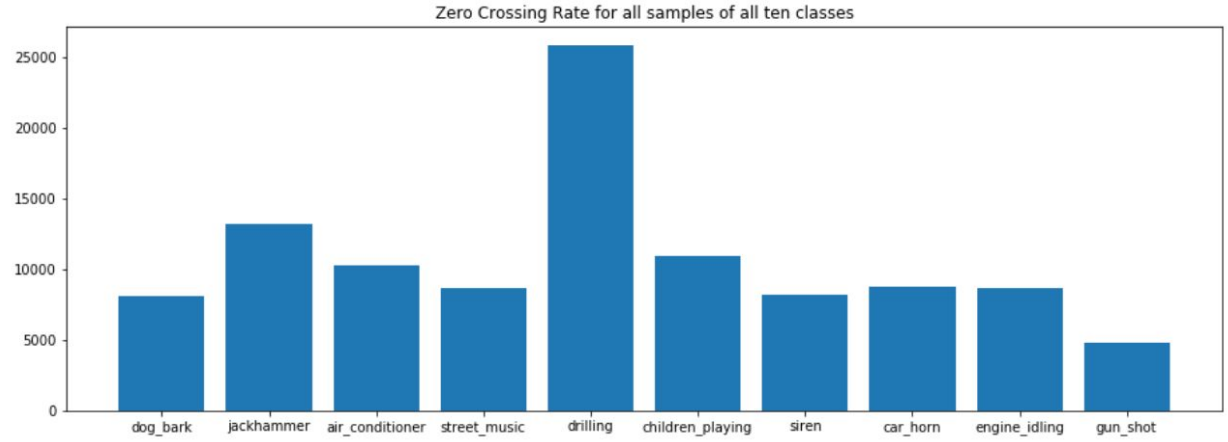

Figure 4: Mean Zero Crossing Rate

Zooming in on a signal's wave plot in Figure 2, the number of times the signal changes from positive to negative and vice-versa can be quantized as the zero-crossing rate, which is a feature used heavily in

sound and music classification. Figure 4 shows the computed mean zero crossing rates for all samples in the dataset. Looking at the wave plots and spectrograms in Figures 2 and 3, the “children playing” spectrogram has the highest rate of change i.e. signal variability, while “engine idling” is the least variable. Just looking at the wave plots of “jackhammer”, “air_conditioner” and “street_music”, one cannot discern any differences. However, their spectrograms tell a different story. The spectrogram of “jackhammer” is almost high and vertical while that of “air_conditioner” is very low. “street_music” is not as steady as either of the former categories but is intermittently louder than “air_conditioner”. “drilling” almost has a horizontal spectrogram, unlike “street_music” and “children_playing”. The spectrogram of “drilling” is closest to that of “car_horn” and “engine_idling”, albeit it is louder than both former. The spectrograms of “gun_shot” and “dog_bark” are very impulse-like as is that of “siren” that has a delayed impulse start but is steady after that, like “jackhammer”. Here, impulse-like refers to the brief time duration the frequencies are present in the signal. “dog_bark” has a lot more random noise in its signal over time than “gun_shot”.

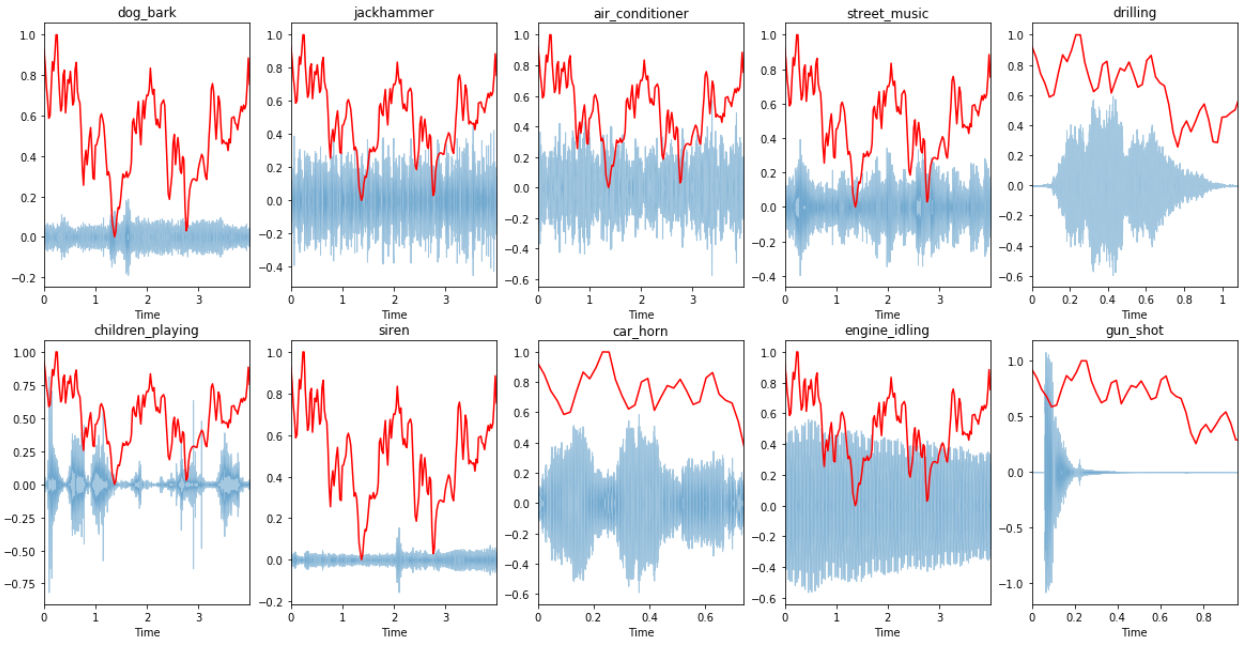

Figure 5: Spectral Centroid of samples

A statistical feature used in sound signal analysis is the spectral centroid. It is defined as the weighted mean of frequencies present in a signal frame. In other words, it could be explained as the center of

mass of the sound signal in each time frame. Just looking at the plots in figure 5, we can cluster them into two groups, one group includes “car_horn”, “gun_shot” and “drilling” and the other group has the rest.

Part 2: Power or Energy Spectrum:

A spectral roll-off is defined as the frequency of the signal at which 85% of the spectral energy is bounded.

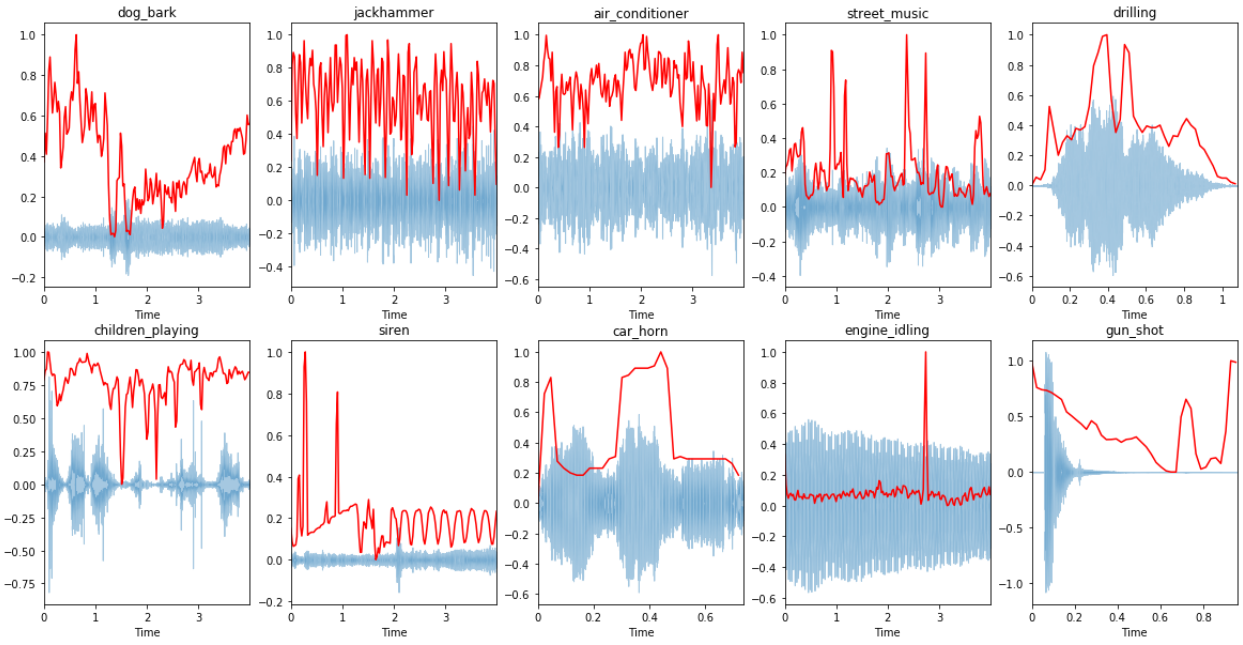

Figure 6: Spectral Roll-off of chosen samples

In Figure 6, the red curves represent the computed spectral roll-offs for each time frame for each sample. The power spectral density of a signal could be represented as a set of discrete coefficients known as the Mel-Frequency Cepstral Coefficients. These coefficients could be used as a feature for sound identification that are mapped to the nonlinear mel-scale that approximates the human auditory system.

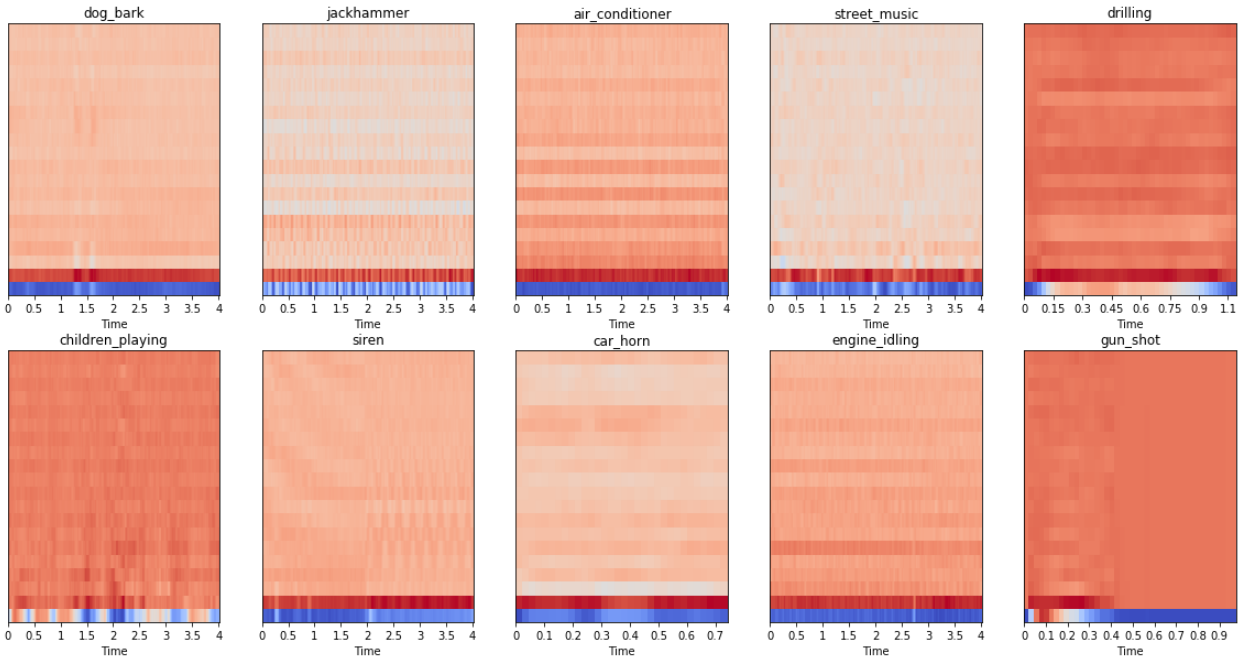

Figure 7: MFCCs for each chosen sample

Figure 7 shows a representation of 20 MFCC features computed for frames of each sample.

In this subsection, we have explained and visualized some features most commonly used in sound signal analysis. We computed these features on a sample of signals from our training dataset. By visualizing the sound data, we could discern distinguishable features between sound classes. In the next subsection, we will express these findings in a more statistical framework.

3.4.2. Inferential Statistics

Following up from the data wrangling step, we will explore the dataset from a statistical point of view. As a review, the dataset is composed of 5435 sound files indexed by a numeric ID that we want to classify into ten classes or types of urban sounds. Ground truth labels are stored in a dataframe called labels. To start, we are interested to know how many total files are in each class. After grouping and counting ids, we get the following table (Table 1) of the number of files present in the dataset for each of the ten classes.

Table 1: Number of files in each class

Class	Number of file samples
air_conditioner	600
car_horn	306
dog_bark	600
children_playing	600
drilling	600
engine_idling	624
gun_shot	230
jackhammer	668
siren	607
street_music	600

We will conduct our statistical analysis through two separate fields of view. First, we will treat the sample space as consisting of the different sound files. In this view, all samples are independent, and we can apply classical statistical methods in our analysis. The second view we implement is the time series modeling approach where we model our sound amplitude or frequency data as observations from a stochastic process. What follows is a description of both views in two separate parts.

Part 1: Independent Samples:

Looking at the data storytelling plots from before, we come up with the following hypothesis: the means of “jackhammer”, “air_conditioner” and “street_music” wave plot (amplitude vs time) sample distribution is the same. In other words, we can state the null hypothesis as the difference in sample distribution means of the three classes is zero.

To test this claim, we will reduce the varying amplitudes over time to the mean amplitude for each file sample then apply the t-test on the somewhat independent mean samples of each class. We run this t-test three times to compare the means of the three class distributions.

Claim 1: The difference in sample distribution means of “jackhammer”, “air_conditioner” and “street_music” classes is zero.

Tests and Results:
1. Test: ttest_ind(jackhammer_mean_files, air_conditioner_mean_files)
Result: t-statistic = 2.8805317026359107, p-value = 0.004036935822494086
Since the resultant p-value is less than 0.05 (5%), we can reject the null hypothesis of equal distribution means.
2. Test: ttest_ind(jackhammer_mean_files, street_music_mean_files)
Result: t-statistic = -1.013447040166551, p-value = 0.31104032436124646
Since the resultant p-value is greater than 0.05 (5%), we can neither reject nor accept the equal means hypothesis. The two groups might have equal means.
3. Test: ttest_ind(street_music_mean_files, air_conditioner_mean_files)
Result: t-statistic = 2.5641908015042962, p-value=0.010462734082099105
Since the resultant p-value is less than 0.05 (5%), we can reject the null hypothesis of equal distribution means.

Part 2: Time Series Approach:

We will answer the following questions in this section:
1. Investigate the stationarity of “street_music” and “siren” using Augmented Dickey Fuller Test. (Null Hypothesis is that data are non-stationary)
2. Are “jackhammer” and “air_conditioner” more stationary than “street_music”?
3. How similar (or correlated) are the wave plots of “drilling”, “engine_idling” and “car_horn”?

A dataset is stationary if the mean is constant over all time points. If a time series has a unit root, it shows a systematic pattern that is unpredictable. The Augmented Dickey-Fuller Test [James G. MacKinnon, 2010 "Critical Values for Cointegration Tests, Economics Department, Queen's University] is used to test for a possible unit root or the stationarity of a time process. The critical values for the test are as follows for 1%, 5% and 10% confidence levels.
{'1%': -3.4305546957800765,
'5%': -2.86163047020448,

'10%': -2.5668181544937676}

Before we run any statistical tests, we generate a representative wave plot sample for each class distribution by using the mean amplitude of all samples at a time point as the amplitude value of the class at that same time point. Since not all file samples have the same time duration, we do the averaging for the smallest time period present in the class sample files. Table 2 shows the final number of data points computed for each class.

Table 2: Number of data values in the final sample for each class

Class	Number of data values in the representative sample
air_conditioner	97920
car_horn	2205
dog_bark	5284
children_playing	55125
drilling	18383
engine_idling	33805
gun_shot	7333
jackhammer	18720
siren	26001
street_music	32000

Next, we make the following claims:

Claim 2: There is a high degree of stationarity in the wave plots of “street_music” and “siren”.

Tests and Results:

1. Test: adfuller(street_music_rep_sample)
Result: Statistic = -21.61937687757208, p-value = 0.0
Since the resultant p-value is less than 0.05 (5%), we can reject the null hypothesis that the “street_music” data are non-stationary and there is no trend in the time series.
2. Test: adfuller(siren_rep_sample)
Result: Statistic = -18.827392026204237, p-value = 2.022129131934057e-30
Since the resultant p-value is less than 0.05 (5%), we can reject the null hypothesis that the “siren” data are non-stationary and there is no trend in the time series.

Claim 3: The wave plots of “jackhammer”, “air_conditioner” and “street music” are stationary.

Tests and Results:
1. Test: adfuller(jackhammer_rep_sample)
Result: Statistic = -13.869743769856765, p-value = 6.497786183782305e-26
Since the resultant p-value is less than 0.05 (5%), we can reject the null hypothesis that the “jackhammer” data are non-stationary and there is no trend in the time series.
2. Test: adfuller(air_conditioner_rep_sample)
Result: Statistic = -18.912595716668758, p-value = 0.0
Since the resultant p-value is less than 0.05 (5%), we can reject the null hypothesis that the “air_conditioner” data are non-stationary and there is no trend in the time series.
3. Test: adfuller(street_music_rep_sample)
Result: Statistic = -21.61937687757208, p-value = 0.0
Since the resultant p-value is less than 0.05 (5%), we can reject the null hypothesis that the “street_music” data are non-stationary and there is no trend in the time series.

Claim 4: The wave plot of “engine_idling”, “drilling” and “car_horn” are stationary.

Tests and Results:
1. Test: adfuller(engine_idling_rep_sample)
Result: Statistic = -12.348556688899007, p-value = 5.896139209666792e-23
Since the resultant p-value is less than 0.05 (5%), we can reject the null hypothesis that the “engine_idling” data are non-stationary and there is no trend in the time series.
2. Test: adfuller(drilling_rep_sample)
Result: Statistic = -17.573260000963817, p-value = 4.0415103709846376e-30
Since the resultant p-value is less than 0.05 (5%), we can reject the null hypothesis that the “drilling” data are non-stationary and there is no trend in the time series.

3. Test: adfuller(car_horn_rep_sample)
Result: Statistic = -6.937221617019703, p-value = 1.0471009403741136e-09
Since the resultant p-value is less than 0.05 (5%), we can reject the null hypothesis that the "car_horn" data are non-stationary and there is no trend in the time series.

Next, we are interested in finding the correlation between the classes. To test this, we compute the Pearson Correlation Coefficient between the representative samples of the classes in interest. We make the following claim:
Claim 5: The wave plots of "engine_idling" and "drilling" are highly correlated.

Test and Result:
1. Test: stats.pearsonr(engine_idling_rep_sample[:com_len], drilling_rep_sample[:com_len])
Result: Correlation coefficient = -0.0012251173053355313, p-value = 0.8680822072537137
The correlation results indicate that the "engine_idling" and "drilling" classes are not correlated. Since, the p-value is higher than 0.05 (5%), we cannot reject the null hypothesis that there is no correlation between the two sample classes.

Following our data exploration work, we are interested to examine the linear relationship between the wave plots of two sets of classes: "dog_bark", "gun_shot" and "children_playing", "street_music". We end our analysis by investigating that relationship using the Pearson Correlation measure as before.

Question 1: What is the correlation between the wave plots of "dog_bark" and "gun_shot"?

Test and Result:
Correlation coefficient = 0.05881049379426529, p-value = 1.8884083840078154e-05

Question 2: What is the correlation between the wave plots of "children_playing" and "street_music"?

Test and Result:
Correlation coefficient = -0.0009760590800338108, p-value = 0.8613971097374443

The first result indicates a very small correlation (close to zero) between "dog_bark" and "gun_shot" that is statistically significant due to the low p-value (< 0.05), while the second result is not statistically significant at all.

In this subsection, we applied inferential statistics techniques to our dataset after the data exploration stage. We applied various statistical tests in two viewpoints with different assumptions to wave plot data

i.e. amplitude values versus time. It would also be interesting to apply similar inference techniques to 2D spectrogram data i.e. amplitude values versus frequencies and time. However, this requires some independent component analysis of the spectrogram matrix to find the independent frequency and time components. Non-negative Matrix Factorization is a popular technique that can be used for this goal.

3.5. Classification Analysis

In this subsection, we build classification models to identify the sound source given a brief audio excerpt. We start by building our space of features from the digitized audio samples, then use these features to train classifier models that can learn patterns from the sound data to correctly identify the class of the sound source.

3.5.1. Feature Extraction

We start building our feature space by computing certain characteristics of the input sound signals. The features we use to build our classifier are the zero-crossing rate, the spectral centroid and roll-off of the signal in addition to 20 mel-frequency cepstral coefficients that describe the signal's power spectrum. Please refer to the Data Exploration and Storytelling report for more details on the description of the extracted features. We compute features for the whole dataset of 5435 samples encompassing the ten classes of sound sources and store them in a pandas dataframe that we can save as a csv file. To facilitate building our classifier in the next step, we merge the labels dataframe loaded from the train.csv file that contains the class labels for each sound sample file with the computed features dataframe using the file ID as a joining key. The resultant merged dataframe can be saved to use in the following classification step.

3.5.2. Classification Models

Before fitting our classification model, we split the dataset into two independent training and testing subsets. We use a training to testing ratio of 80:20 samples. Next, we encode all ten class labels as categorical target variables. We also normalize all feature values by subtracting the mean and scaling to unit variance.

We start the fitting process by building a random forest classification model that is an ensemble of 100 single decision trees. We find the tree nodes that discriminate the features and split the feature space by minimizing the gini purity index for each node. A random subset of features is used for every iteration to find the tree nodes. We then use the learned ensemble trees along with the test subset to predict class labels for the test samples.

We also use a k-NN classifier with 10 neighbors to classify test samples using the majority vote of the smallest distanced neighbor to the sample in the feature space.

4. Model Performance

To assess and compare the performance of both classifiers, we will compute the mean classification accuracy score, the precision, recall, F1-scores and Matthew's Correlation Coefficient (MCC) for each independently. Results were as shown in Table 3.

Table 3: Classification Results for both classifiers

Classifier	Random Forest Tree (100 estimators)	k-Nearest Neighbor (k=10)
Mean Accuracy Score	0.9	0.83
Precision	0.9	0.83
Recall	0.88	0.83

F1-Score	0.89	0.82
MCC	0.88	0.81

We also ran cross validation tests to better represent the models' ability to generalize. Mean Accuracy scores for 5-fold cross validation tests are shown in Table 4.

Table 4: Cross-validation (k=5) Accuracy Scores for both classifiers

	Random Forest Tree (100 estimators)	k-Nearest Neighbor (k=10)
Mean Accuracy Score	0.91	0.84

From Table 3 and Table 4, we can see that the bootstrap aggregation (bagging) process of the 100 individual trees along with randomizing the feature selection process produced superior results to just classifying new data samples according to their 10 nearest data points.

5. Conclusion

In this project, we developed classifier models capable of identifying sound sources from a labeled set of sound data. We applied traditional data analysis techniques: we started by acquiring the data followed by reformatting the data to fit our modeling process. Using spectrograms and waveplots we were able to explore the data for interesting features. We ran inferential statistical analysis by testing hypotheses from independent samples and time series approaches. We then used the extracted interesting features to train two different types of classifiers: a simple K-Nearest Neighbor classifier and an ensemble decision tree

classifier (Random Forest Modeling). We were able to achieve good classification accuracy on unseen sound samples and as expected the Random Forest classifier scored higher on all computed test metrics.