Springboard DSC

CAPSTONE PROJECT 2

URBAN SOUND CLASSIFICATION

Final Report

Vikram Lucky

Dr.vlucky@gmail.com

16 May 2019

Description and Objectives

We all get exposed to vast variety of urban sounds every day. Unlike speech and audio signals, Urban sounds are usually unstructured sounds. They include various real-life noises generated by human activities ranging from car horns, engine idling to music and children playing. The aim of this work is to extract features from audio files and using those features create Machine Learning (ML) models to accurately label such audio files into their specific categories (*out of 10 categories, in this case*). The idea is to discover relevant patterns in the audio features and use the discovered patterns as characterizing attributes for ML models. In this project we used machine learning algorithms like Logistic Regression, Decision trees, XGBoost, and Deep Neural Networks. We compared the results of these models, and picked the one that most accurately labeled sounds into categories.

The Data

Publicly available dataset called "*UrbanSound-8k*", created by Justin Salamon, Christopher Jacoby, and Juan Pablo Bello is used for this project. This dataset is a collection of 8,732 short audio clips (upto 4 secs) of urban sound areas. The dataset is divided into 10 classes: *air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren and street music.*

Data Wrangling

As with all unstructured data formats, a data has a couple of preprocessing steps which have to be followed before it is presented for analysis. The first and most important step is to extract features from audio files, so it is ready for machine learning analysis. We used library called "*LibROSA*" for feature extraction. LibROSA is a python package for music and audio analysis. It provides the building blocks necessary for to create audio information retrieval systems. LibROSA provides several methods to extract different features from sound clips. For this project we used the following methods to extract various features:

- *melspectrogram: Compute a Mel-scaled power spectrogram*
- *mfcc: Mel-frequency cepstral coefficients*
- *Spectral_contrast: Compute spectral contrast*
- *Chroma: Compute a chromagram from a waveform or power spectogram*
- *Tonnetz: Compute the tonal centroid features*

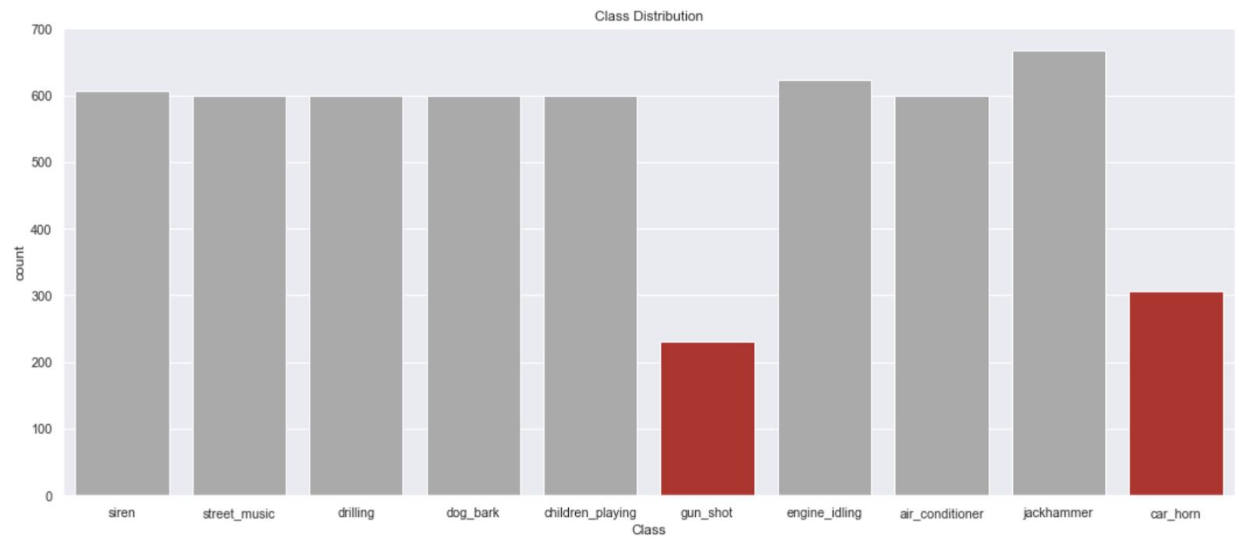
A Python function was written which went over each audio file present in the provided training set and extracted above mentioned features from each one of them and it returned:

- *Tonnetz* data frame (5435, 7)
- *Mfccs* data frame (5435, 41)
- *Chroma* data frame (5434, 13)
- *Mel Spectrogram* data frame (5435, 129)
- *Contrast* data frame (5435, 8)
- df: A data frame combined of all of the data frames above (5435, 196)

Exploratory Data Analysis and Inferential Statistics

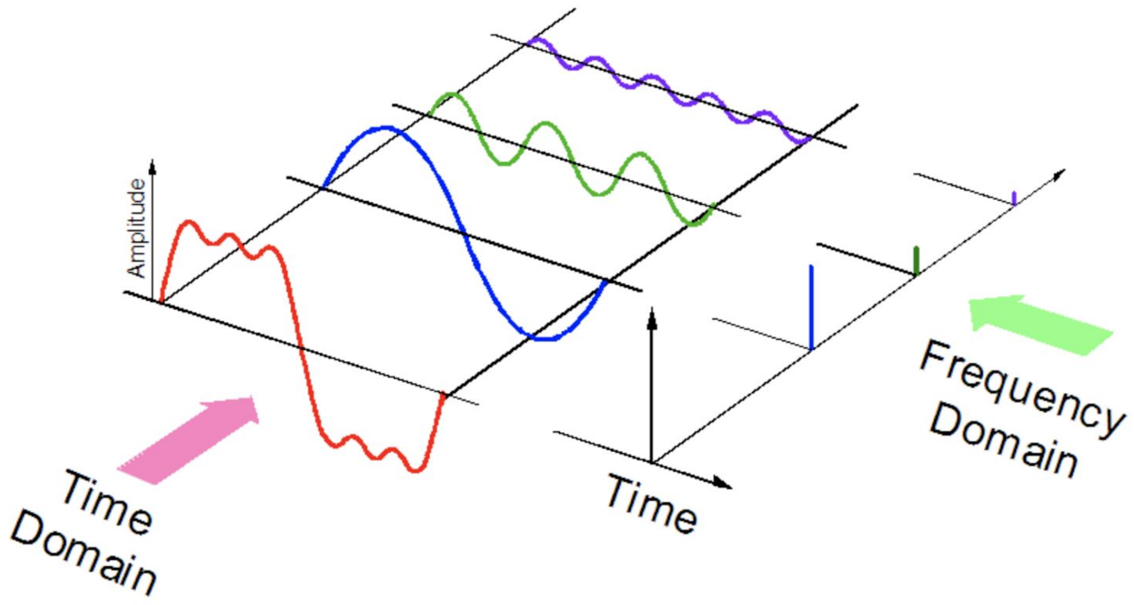
In this section we present a series of questions we posed as part of our storytelling.

We began by looking into the distribution of Classes:



Bar graph above shows distribution of classes. It can be observed that *car_horn* and *gun_shot* are under represented, when compared to other classes present in dataset. As other classes were almost equally distributed, I decided not to balance minority classes by introducing additional external or synthetic data, until I look at the classification report.

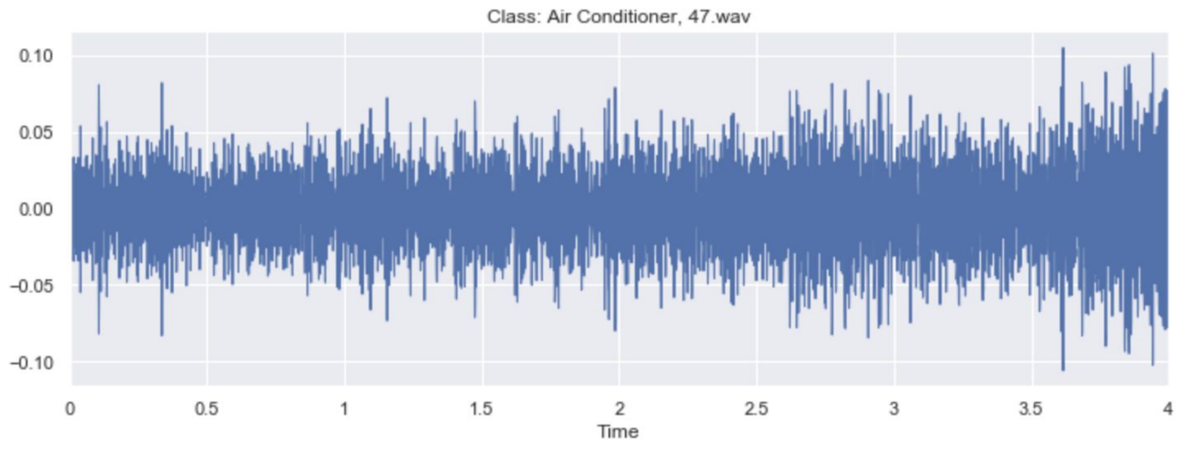
There are 2 commony ways to represent sound:

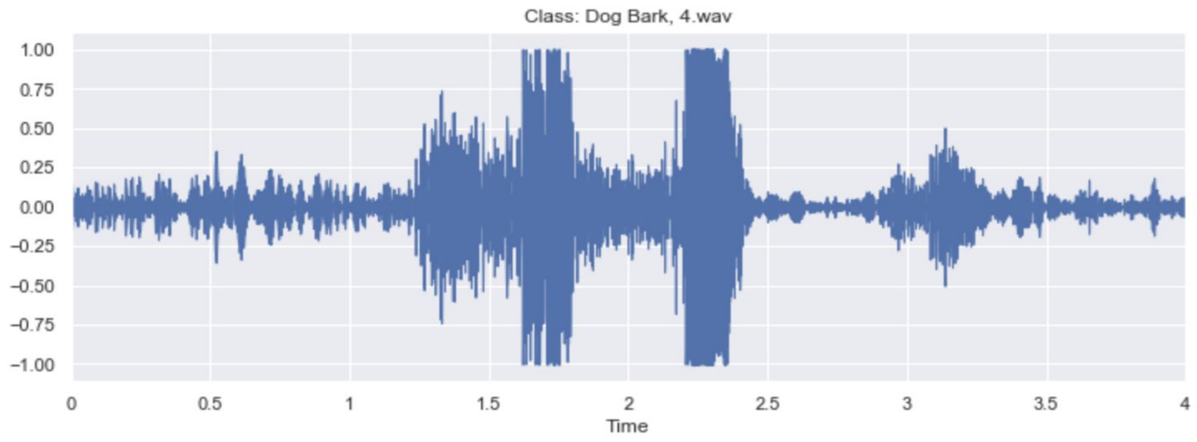


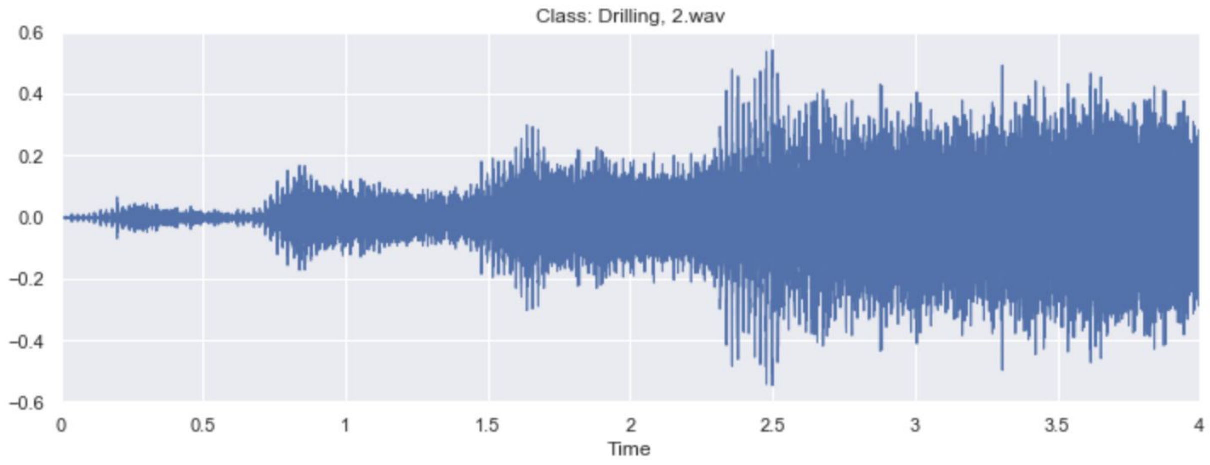
- **Time domain**: each sample represents the variation in air pressure.
- **Frequency domain:** at each time stamp we indicate the amplitude for each frequency.

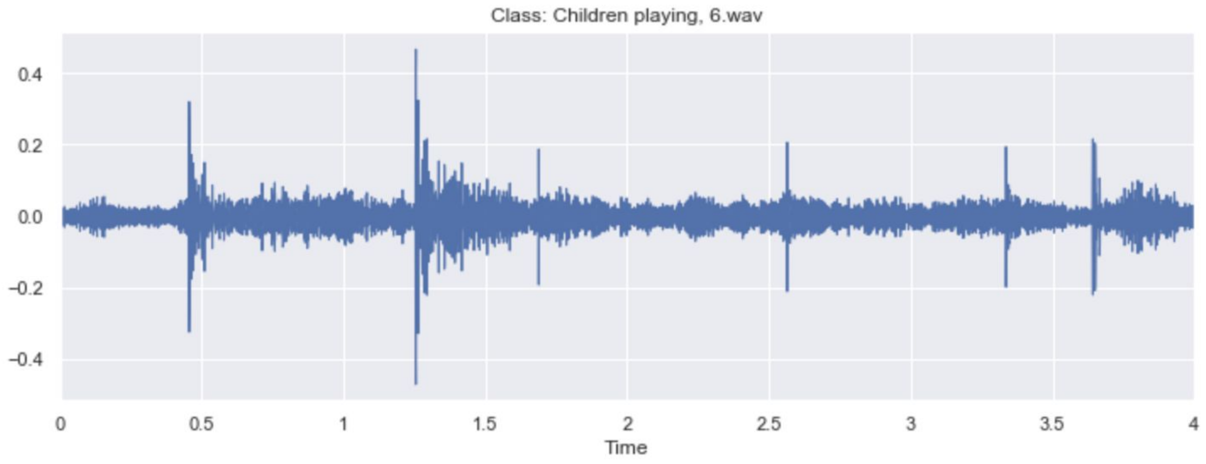
As shown in the above picture, there are 2 common ways to represent audio sound: Time domain and Frequency domain

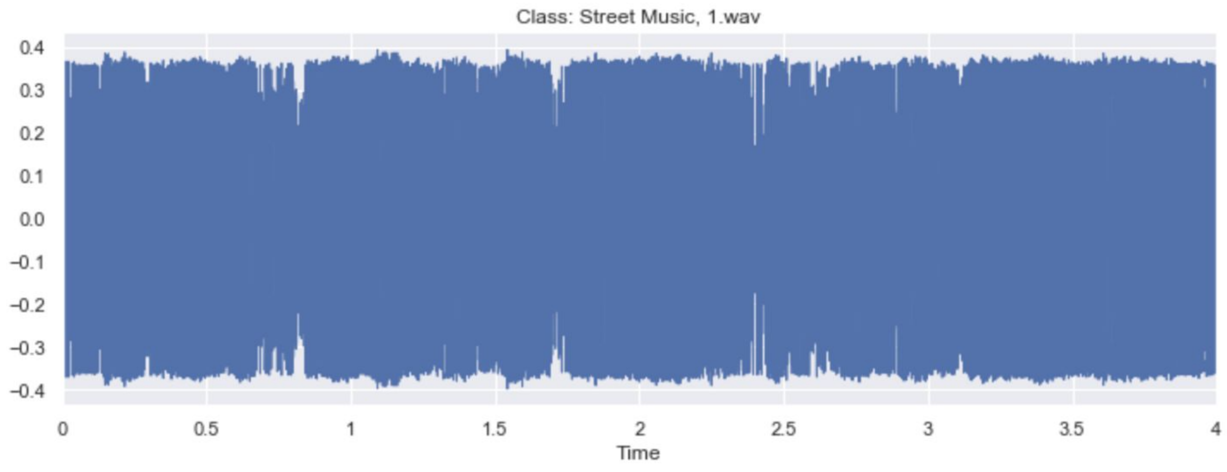
Time domain: wave plots of different class types:

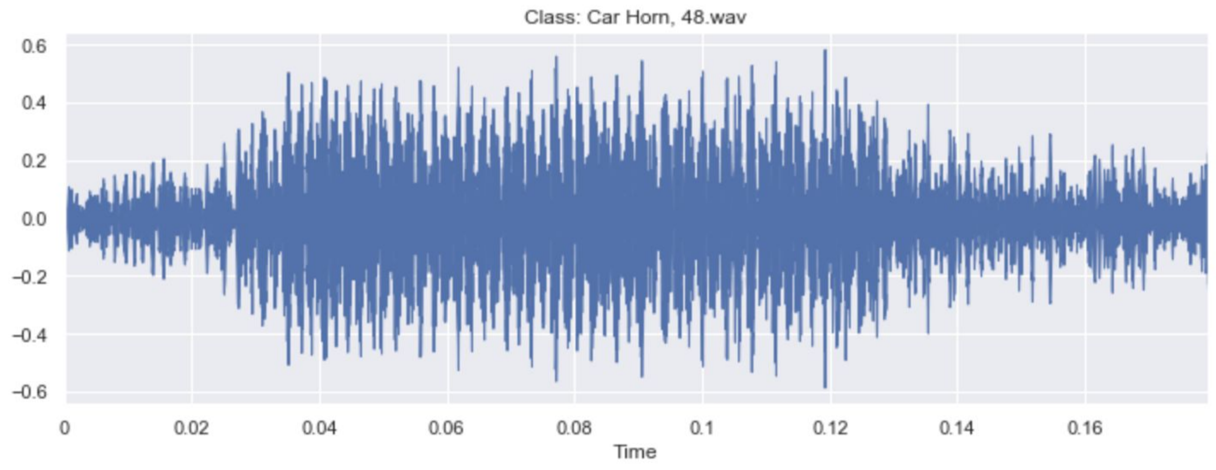


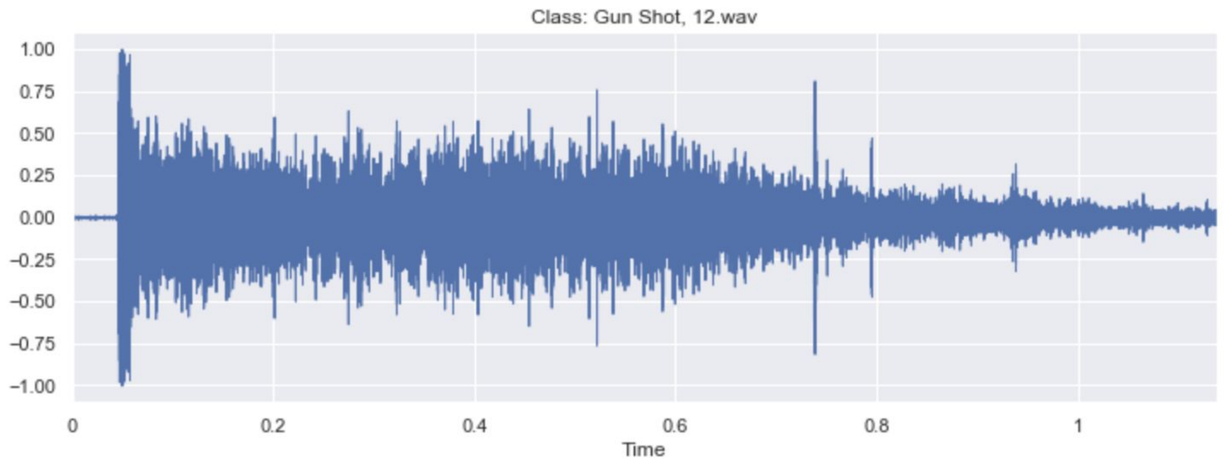

Class: Dog Bark, 4.wav
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
0.5
1
1.5
2
2.5
3
3.5
4
Time


Class: Drilling, 2.wav
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
0.5
1
1.5
2
2.5
3
3.5
4
Time


Class: Children playing, 6.wav
0.4
0.2
0.0
−0.2
−0.4
0
0.5
1
1.5
2
2.5
3
3.5
4
Time

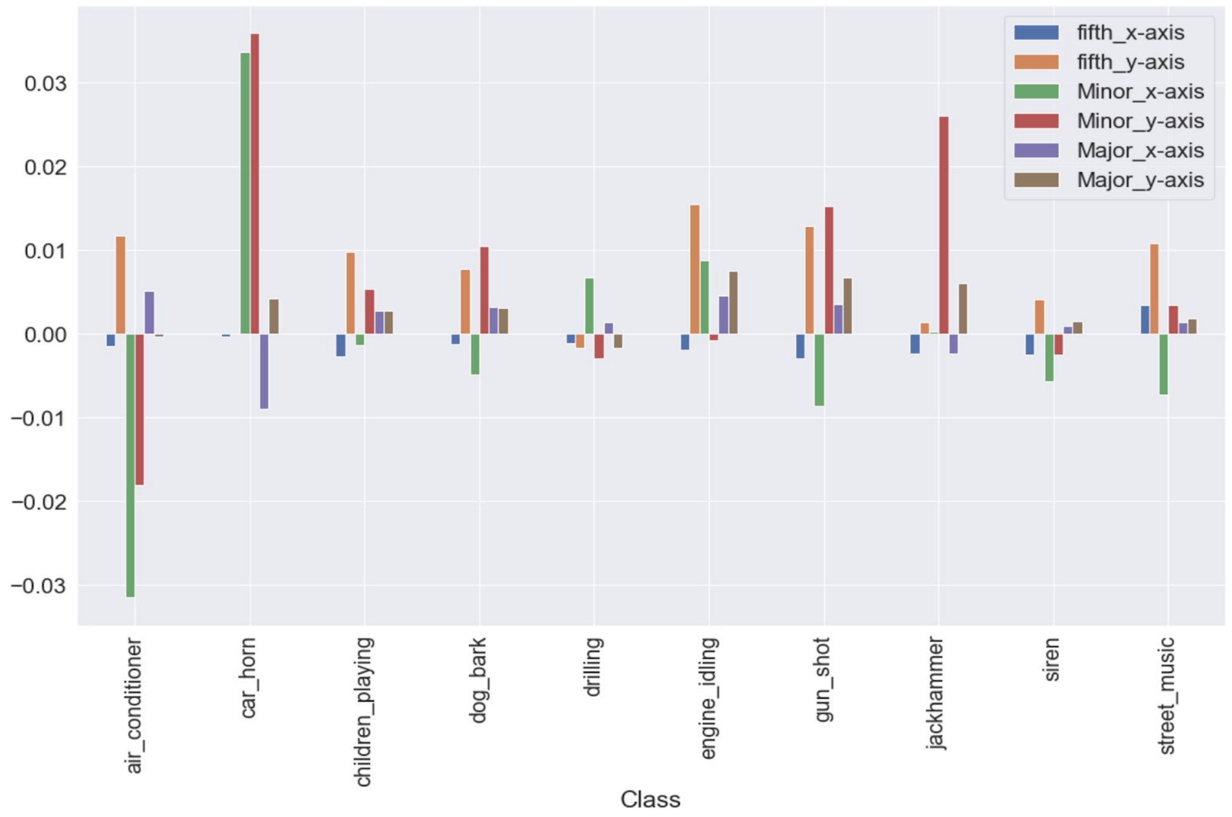






The above waveform plots show the differences in the sound waves for each specific class.

Bar graph of Tonnetz: Tonnetz computes the tonal centroid features.

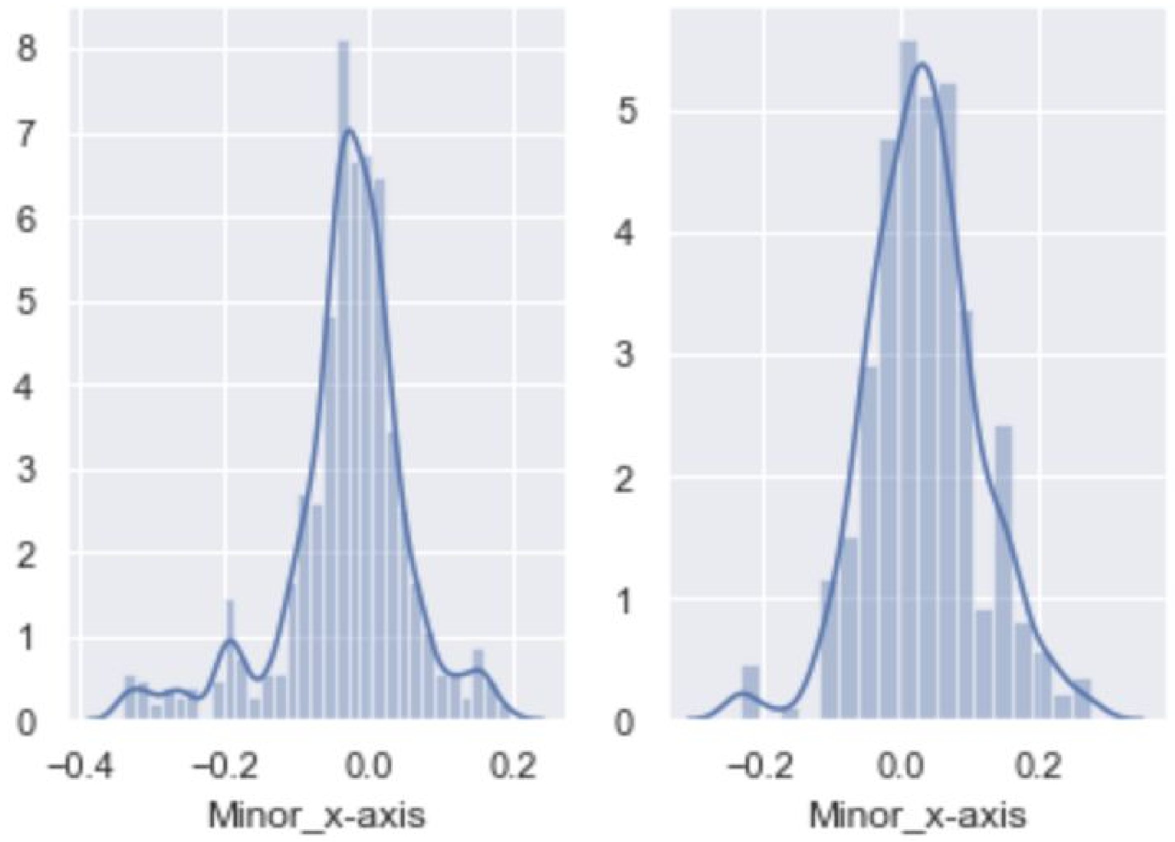


Above graph shows average tonal centroid features for each frame for different classes. A noticeable pattern can be observe between *car horn* and *air conditioner* classes:

- *Average Minor_x-axis and Minor_y-axis is higher for **car_horn** class compare to other classes*
- *Average Minor_x-axis and Minor_y-axis is the lowest for **air conditioner** class compare to all of the other classes.*

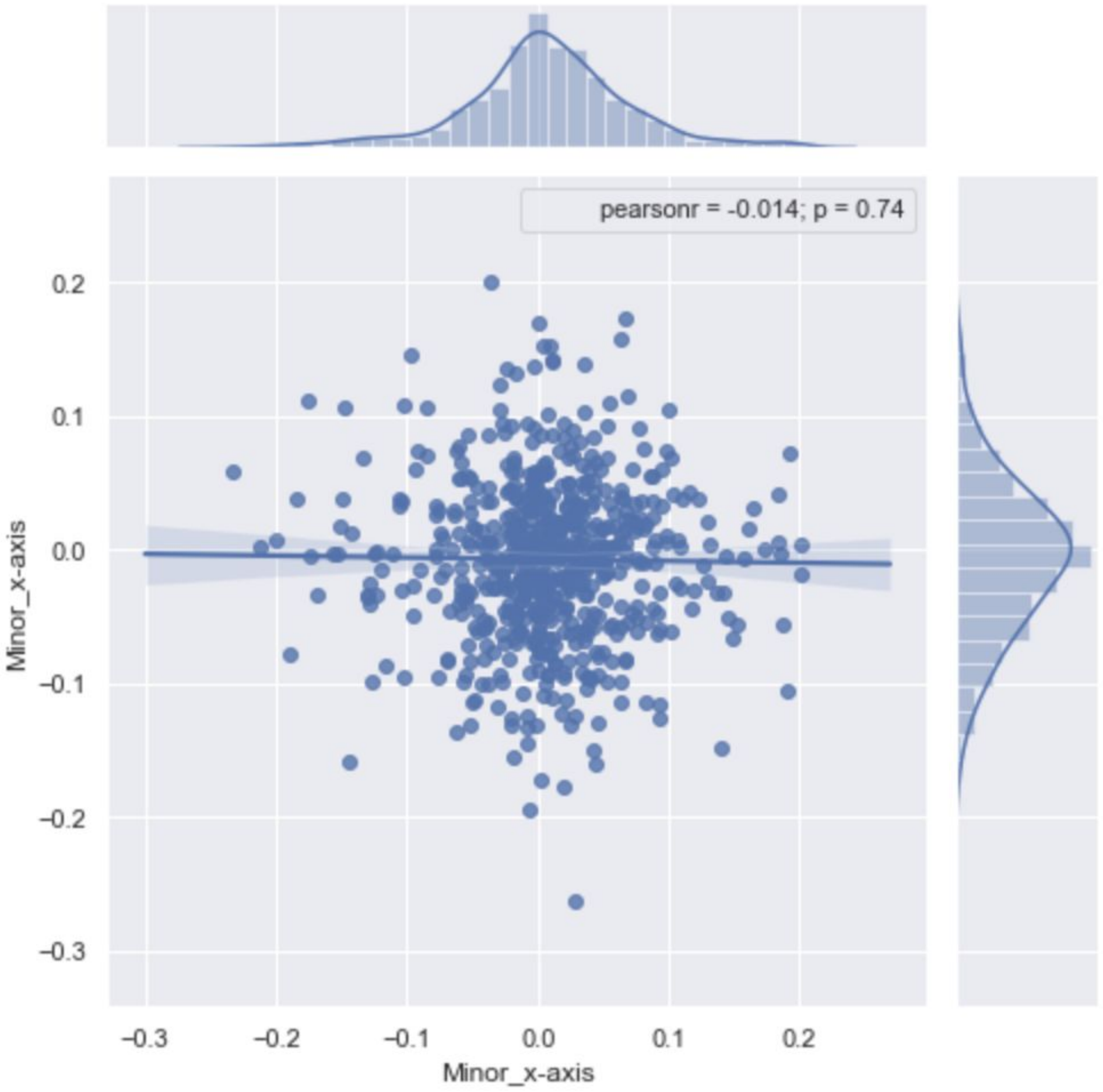
Inferential Statistics:

Hypothesis test was conducted to check the relationship between Minor_x-axis of **car_horn** and **air conditioner** classes. Is difference between their mean statistically significant?



Ho: Difference between car_horn's mean and air_conditioner's mean is not statistically significant
H1: Difference between car horn class's mean and air conditioner class's mean is statistically significant
a: 0.05
P - value observed were less than the threshold, therefore Ho was rejected.

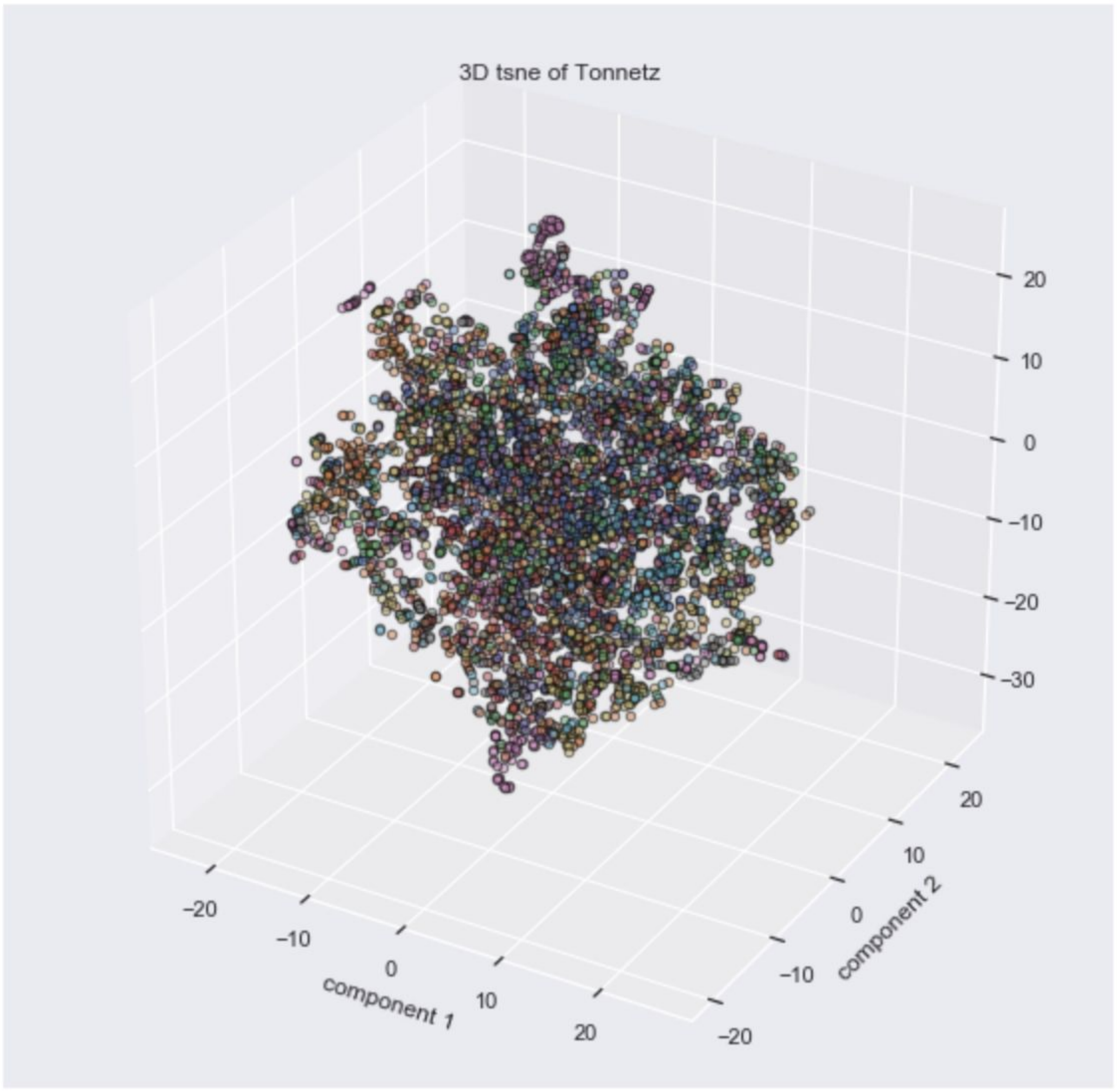
Pearson's correlation coefficient test was conducted to measure the statistical relationship between **Minor_x-axis** of drilling and street music classes.



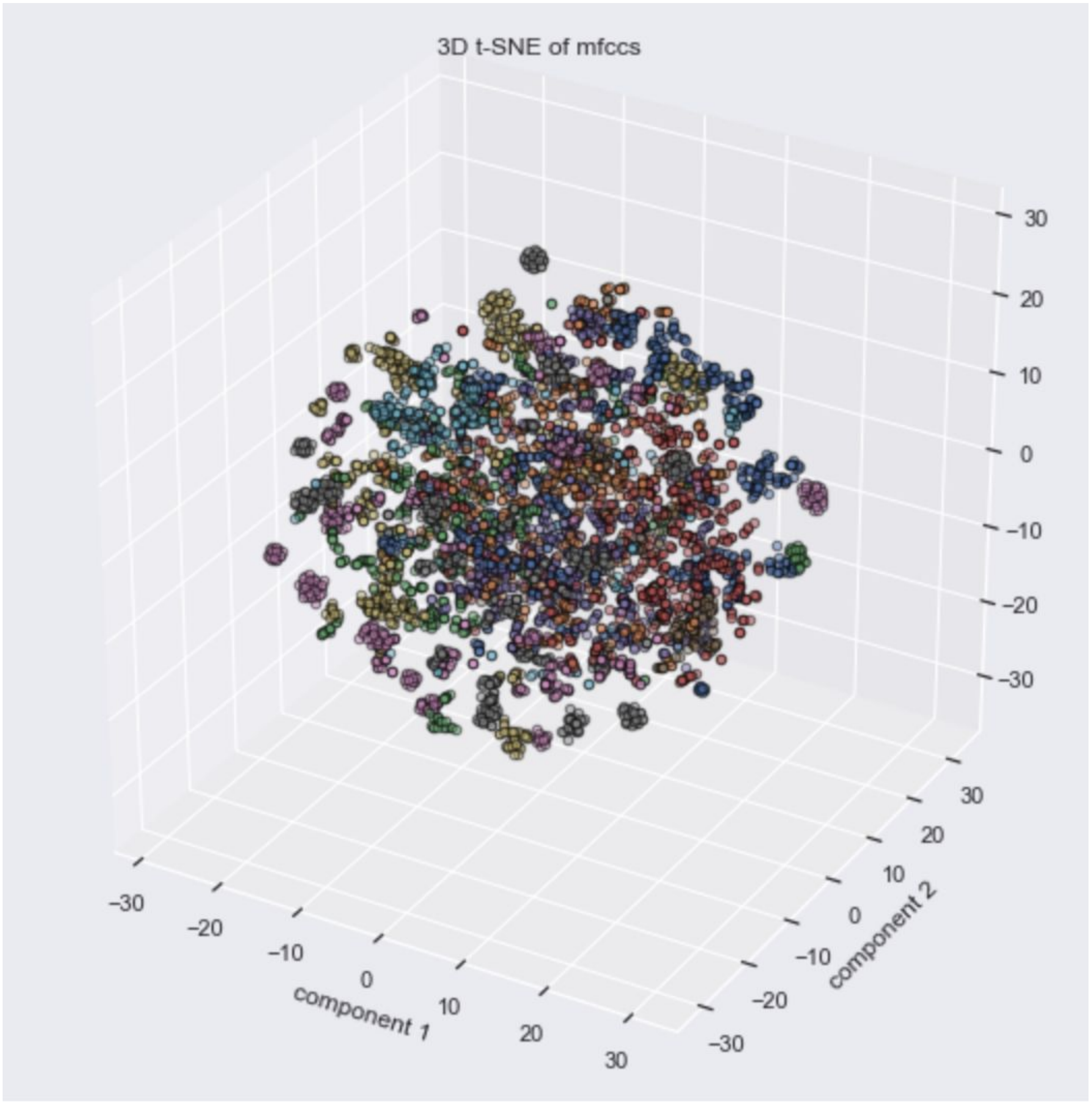
As p value is 0.74, much higher than threshold 0.05. The null hypothesis could not be rejected, therefore we cannot say whether the observed correlation between the two variables is statistically significant or not.

Data Visualization using 3D graphs

As the number of column increases it became harder for us to identify any hidden patterns using basic histograms and bar graphs. Therefore visualization technique like **t-SNE**(t-distributed stochastic neighbor embedding) was used to explore data in 3D space.



Above graphs shows Tonnetz features in 3D space, different colors represents different classes. It’s hard to differentiate visually between classes using Tonnetz.



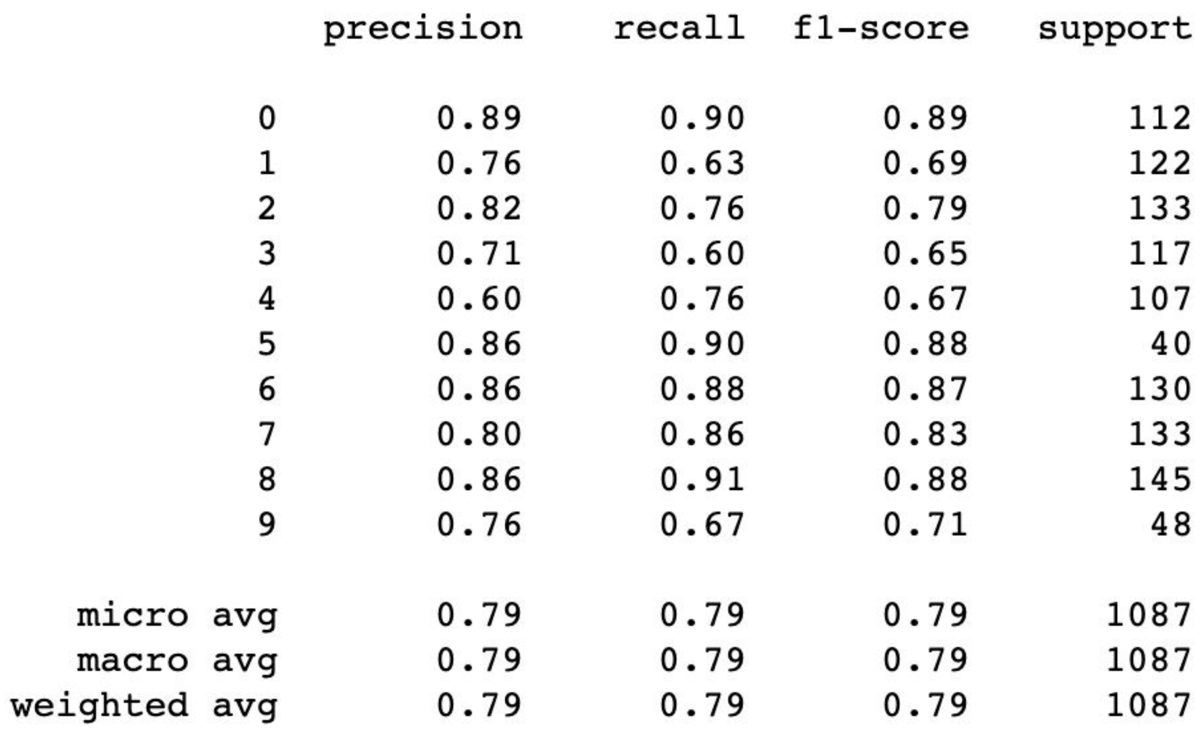
Above graph shows Mfccs features in 3D space, much better than Tonnetz representation of data. As some classes grouped together can be easily observed visually.

PREDICTIVE MODELING

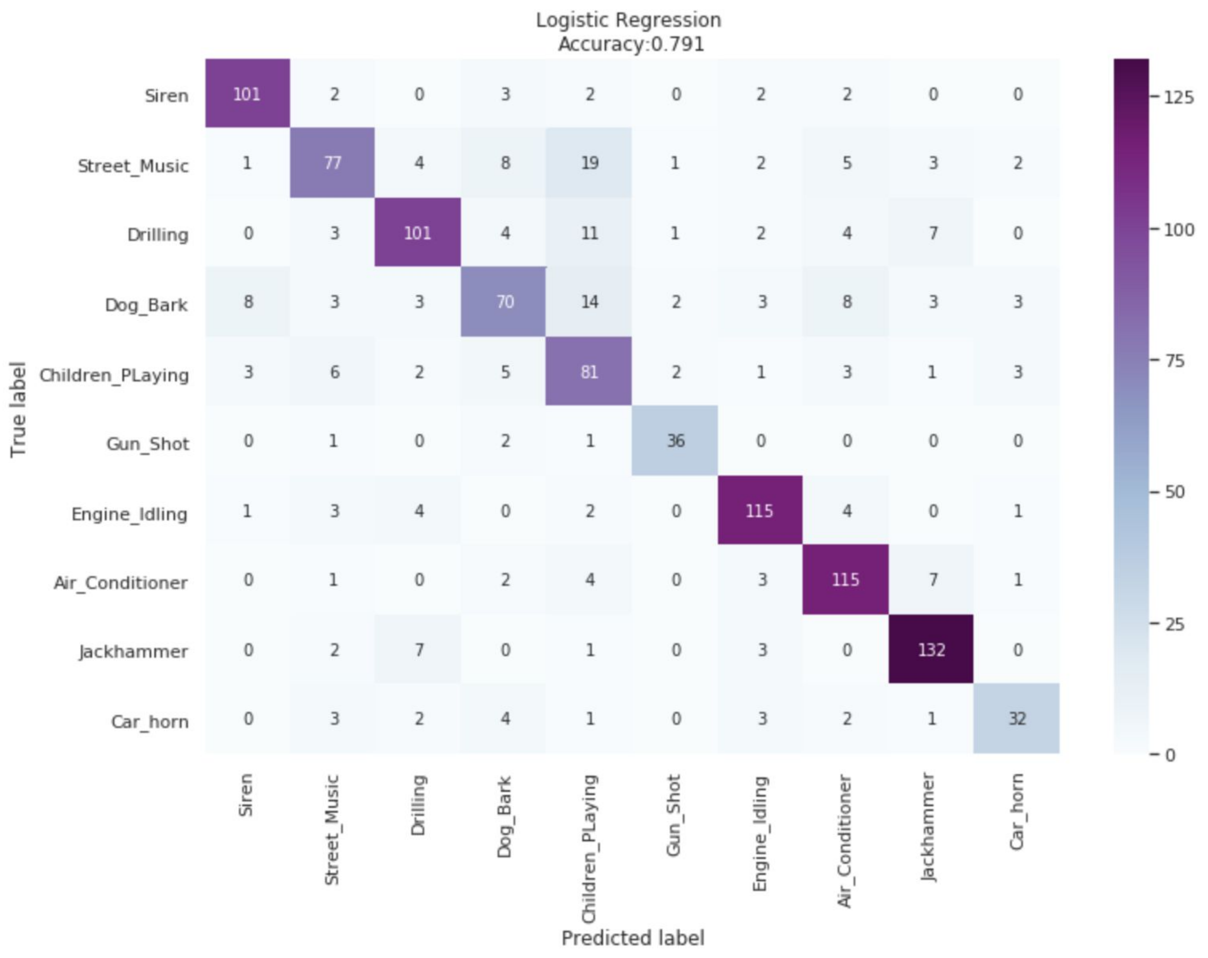
Baseline modeling

In this part of the project, our first task was to create a baseline set of classification experiments to compare our results with. The goal of the baseline experiments was not to produce optimal parameters and to maximize accuracy, but to study the problems and characteristics of the dataset itself.

We used Logistic Regression with various regularization techniques. We ran 5-fold cross validation, and in each fold model was trained on 80% of the data and tested on the rest 20%. Our baseline classifier achieved an average classification accuracy of ~78% averaged across all classes.



```
              precision    recall  f1-score   support

           0       0.89      0.90      0.89       112
           1       0.76      0.63      0.69       122
           2       0.82      0.76      0.79       133
           3       0.71      0.60      0.65       117
           4       0.60      0.76      0.67       107
           5       0.86      0.90      0.88        40
           6       0.86      0.88      0.87       130
           7       0.80      0.86      0.83       133
           8       0.86      0.91      0.88       145
           9       0.76      0.67      0.71        48

   micro avg       0.79      0.79      0.79      1087
   macro avg       0.79      0.79      0.79      1087
weighted avg       0.79      0.79      0.79      1087
```

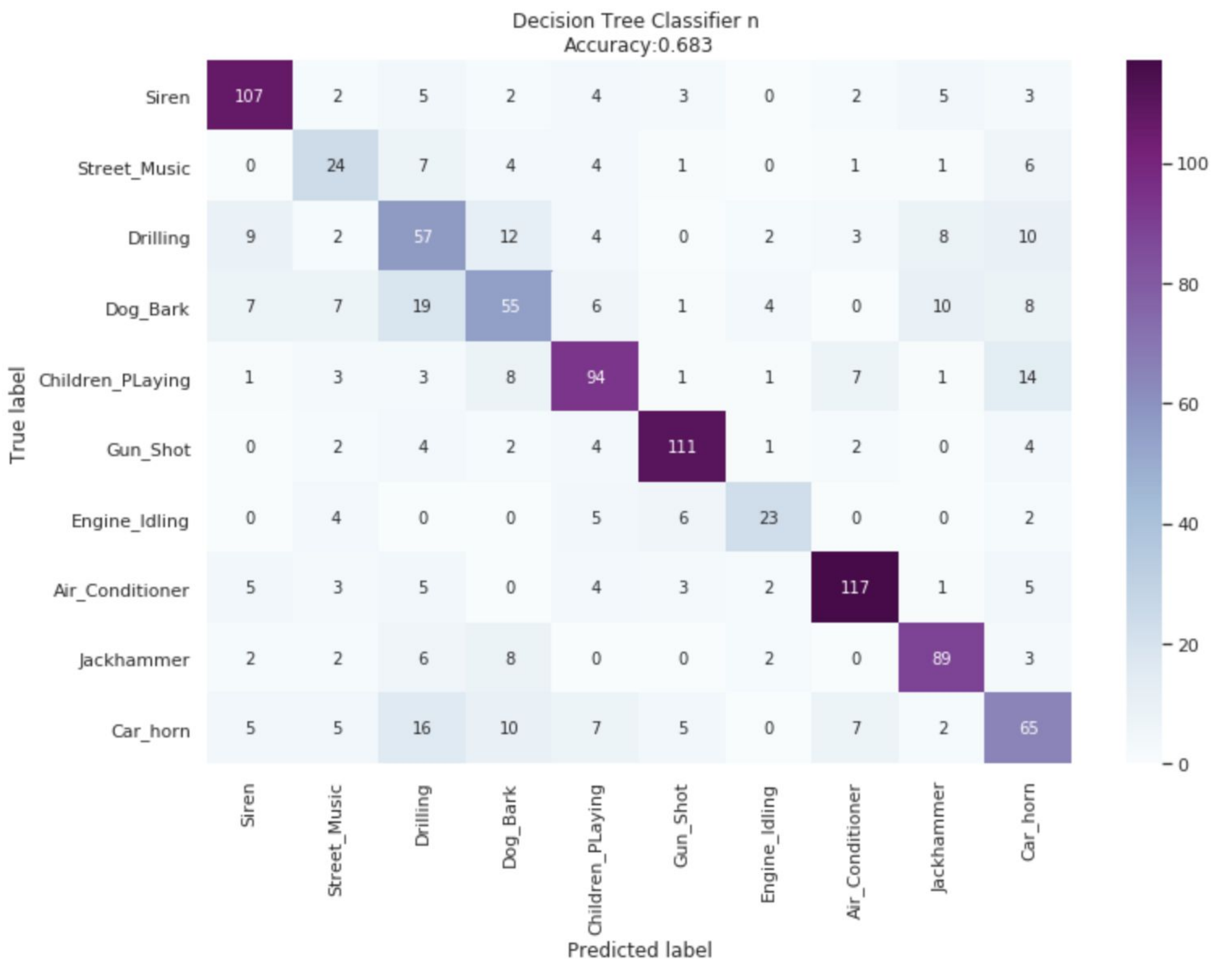


True label \ Predicted label	Siren	Street_Music	Drilling	Dog_Bark	Children_PLaying	Gun_Shot	Engine_Idling	Air_Conditioner	Jackhammer	Car_horn
Siren	101	2	0	3	2	0	2	2	0	0
Street_Music	1	77	4	8	19	1	2	5	3	2
Drilling	0	3	101	4	11	1	2	4	7	0
Dog_Bark	8	3	3	70	14	2	3	8	3	3
Children_PLaying	3	6	2	5	81	2	1	3	1	3
Gun_Shot	0	1	0	2	1	36	0	0	0	0
Engine_Idling	1	3	4	0	2	0	115	4	0	1
Air_Conditioner	0	1	0	2	4	0	3	115	7	1
Jackhammer	0	2	7	0	1	0	3	0	132	0
Car_horn	0	3	2	4	1	0	3	2	1	32

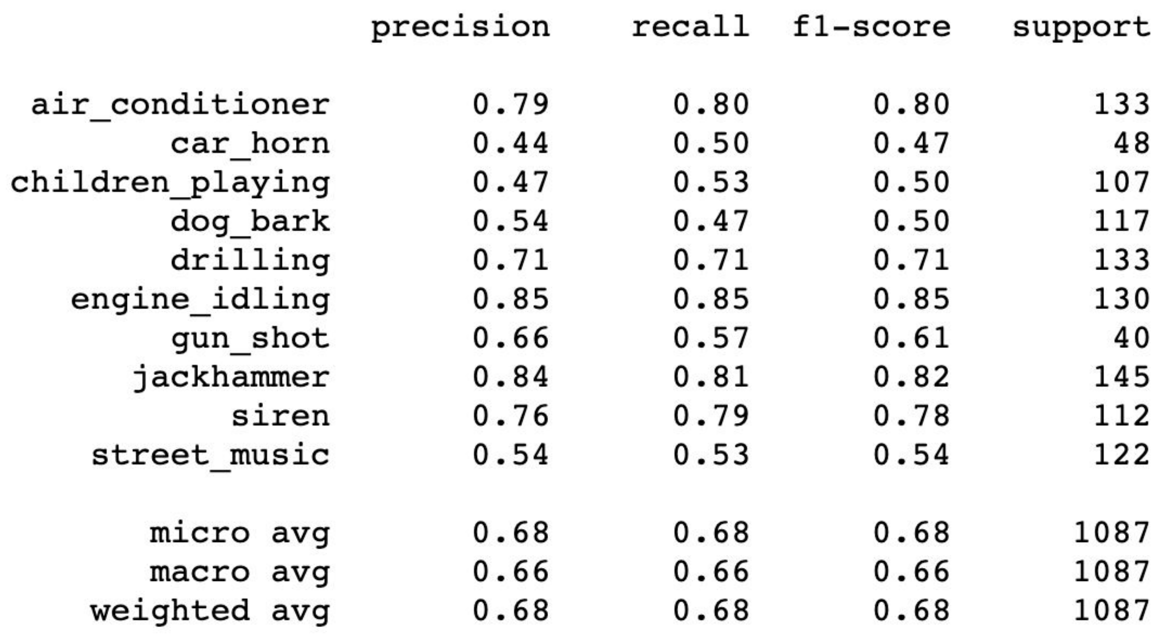
<u>Extended Analysis</u>

In this section we experimented various machine learning models such as: XGBClassifier, Decision Tree Classifier and Neural Networks, and compared their performance. We started with Decision tree classifier and we achieved an average accuracy of 68% among all classes.

Decision Tree Classifier:



True label \ Predicted label	Siren	Street_Music	Drilling	Dog_Bark	Children_PLaying	Gun_Shot	Engine_Idling	Air_Conditioner	Jackhammer	Car_horn
Siren	107	2	5	2	4	3	0	2	5	3
Street_Music	0	24	7	4	4	1	0	1	1	6
Drilling	9	2	57	12	4	0	2	3	8	10
Dog_Bark	7	7	19	55	6	1	4	0	10	8
Children_PLaying	1	3	3	8	94	1	1	7	1	14
Gun_Shot	0	2	4	2	4	111	1	2	0	4
Engine_Idling	0	4	0	0	5	6	23	0	0	2
Air_Conditioner	5	3	5	0	4	3	2	117	1	5
Jackhammer	2	2	6	8	0	0	2	0	89	3
Car_horn	5	5	16	10	7	5	0	7	2	65



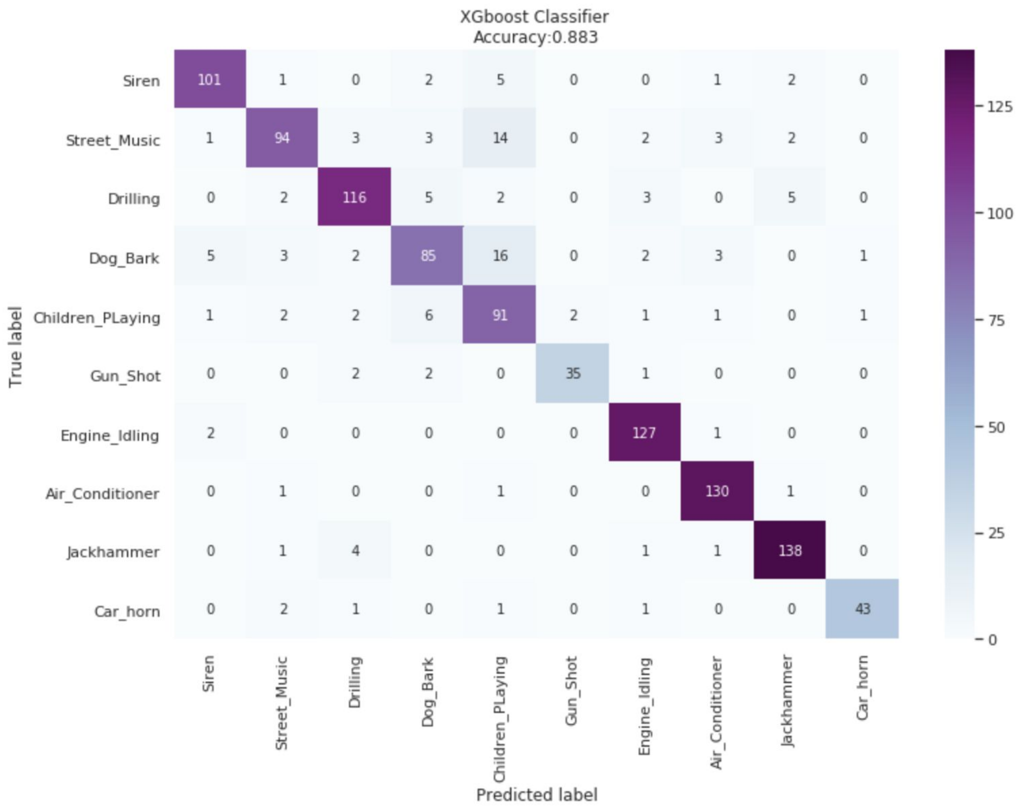
```
                  precision    recall  f1-score   support

 air_conditioner       0.79      0.80      0.80       133
        car_horn       0.44      0.50      0.47        48
children_playing       0.47      0.53      0.50       107
        dog_bark       0.54      0.47      0.50       117
        drilling       0.71      0.71      0.71       133
   engine_idling       0.85      0.85      0.85       130
        gun_shot       0.66      0.57      0.61        40
      jackhammer       0.84      0.81      0.82       145
           siren       0.76      0.79      0.78       112
    street_music       0.54      0.53      0.54       122

       micro avg       0.68      0.68      0.68      1087
       macro avg       0.66      0.66      0.66      1087
    weighted avg       0.68      0.68      0.68      1087
```

Decision tree classifier failed to beat the accuracy achieved by baseline modeling.

<u>XGBoost Classifier:</u>

We also tried XGboost classifier, which is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. XGboost classifier performed really well and it beat baseline model with much higher margin.

```
Accuracy: 88.32%
              precision    recall  f1-score   support

           0       0.92      0.90      0.91       112
           1       0.89      0.77      0.82       122
           2       0.89      0.87      0.88       133
           3       0.83      0.73      0.77       117
           4       0.70      0.85      0.77       107
           5       0.95      0.88      0.91        40
           6       0.92      0.98      0.95       130
           7       0.93      0.98      0.95       133
           8       0.93      0.95      0.94       145
           9       0.96      0.90      0.92        48

   micro avg       0.88      0.88      0.88      1087
   macro avg       0.89      0.88      0.88      1087
weighted avg       0.89      0.88      0.88      1087
```



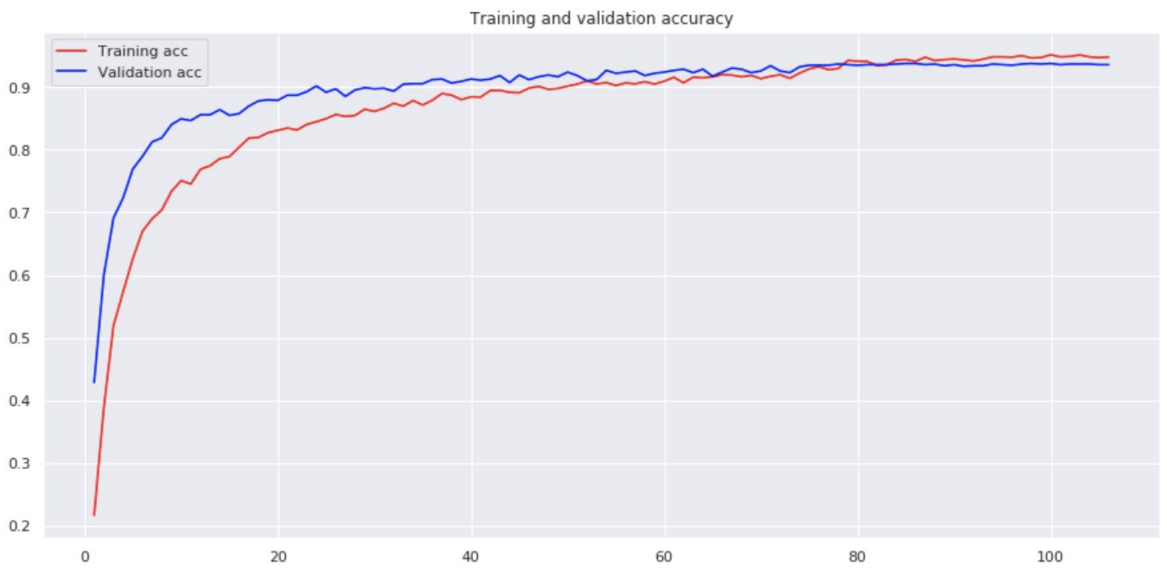
True label \ Predicted label	Siren	Street_Music	Drilling	Dog_Bark	Children_PLaying	Gun_Shot	Engine_Idling	Air_Conditioner	Jackhammer	Car_horn
Siren	101	1	0	2	5	0	0	1	2	0
Street_Music	1	94	3	3	14	0	2	3	2	0
Drilling	0	2	116	5	2	0	3	0	5	0
Dog_Bark	5	3	2	85	16	0	2	3	0	1
Children_PLaying	1	2	2	6	91	2	1	1	0	1
Gun_Shot	0	0	2	2	0	35	1	0	0	0
Engine_Idling	2	0	0	0	0	0	127	1	0	0
Air_Conditioner	0	1	0	0	1	0	0	130	1	0
Jackhammer	0	1	4	0	0	0	1	1	138	0
Car_horn	0	2	1	0	1	0	1	0	0	43

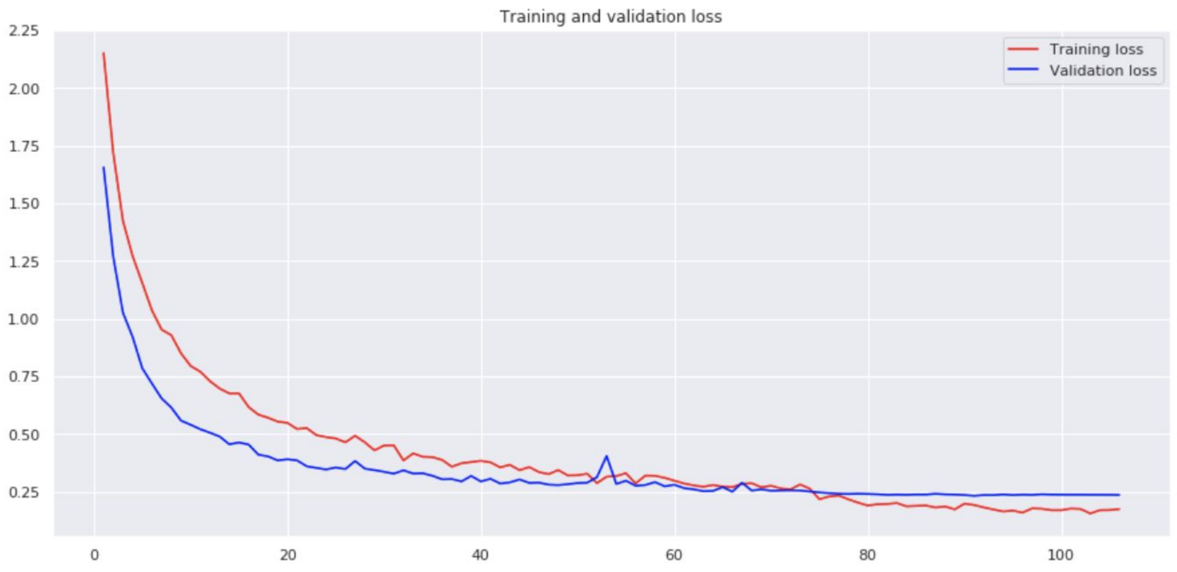
CNN Neural Network:

Xgboost performed really well, Can we beat it's performance using CNN neural networks? To answer this question we created a CNN neural network using Keras as wrapper and with Tensorflow in the backend.

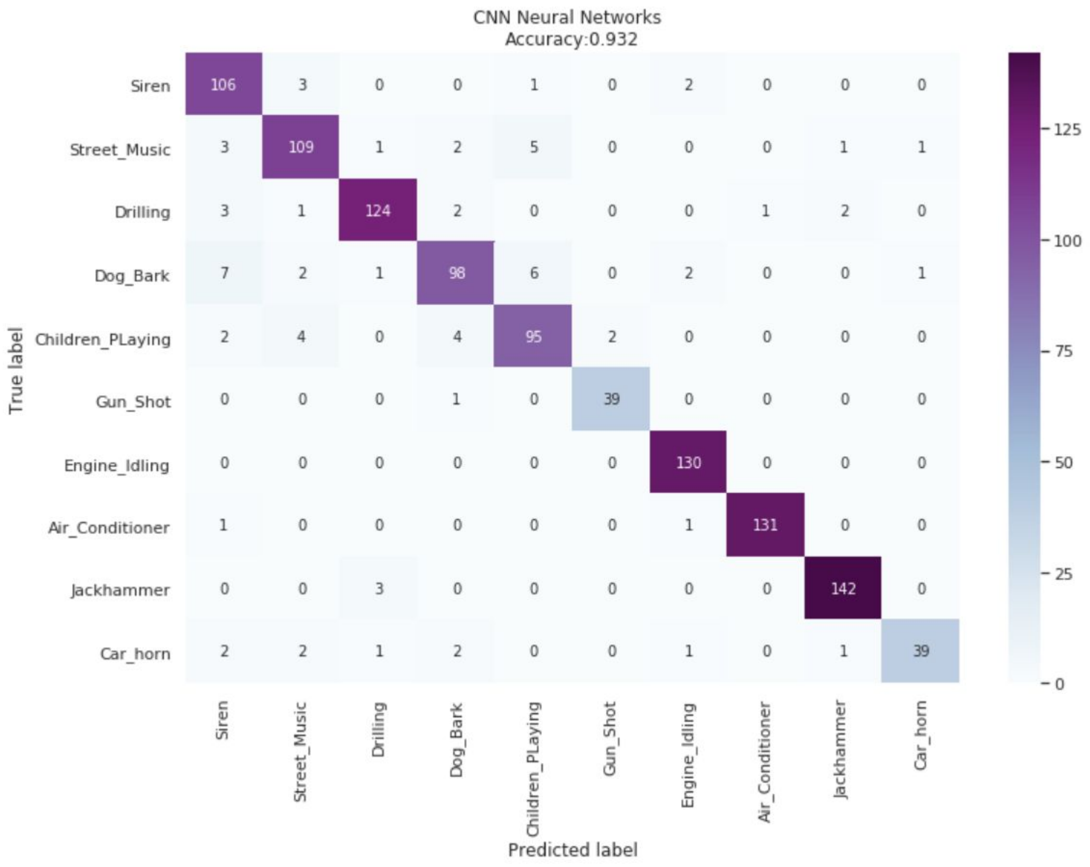
We created CNN model with 5 hidden layers with 'Adam' as the optimizer, an output layer with 'softmax' as the optimizer and a learning rate of 0.001.

- First layer had 120 neurons with 0.5 dropout probability
- Second layer had 256 neurons with 0.5 dropout probability
- Third and fourth layers with 150 neurons and 0.5 dropout probability
- Fifth layer with half the size of input dimension and 0.2 dropout probability
- Output layer had 10 neurons equal to the number of the classes with 'softmax' activation function.


Training and validation accuracy
Training acc
Validation acc
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100

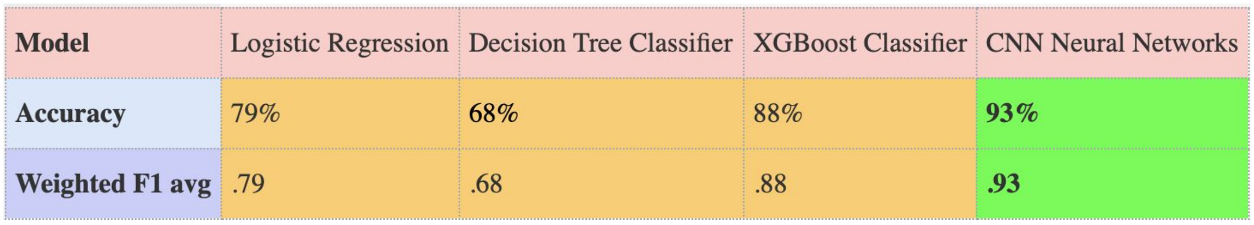

Training and validation loss
Training loss
Validation loss
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
20
40
60
80
100

```
Accuracy: 93.19%
                 precision    recall  f1-score   support

           0       0.85      0.95      0.90       112
           1       0.90      0.89      0.90       122
           2       0.95      0.93      0.94       133
           3       0.90      0.84      0.87       117
           4       0.89      0.89      0.89       107
           5       0.95      0.97      0.96        40
           6       0.96      1.00      0.98       130
           7       0.99      0.98      0.99       133
           8       0.97      0.98      0.98       145
           9       0.95      0.81      0.88        48

   micro avg       0.93      0.93      0.93      1087
   macro avg       0.93      0.92      0.93      1087
weighted avg       0.93      0.93      0.93      1087
```



CNN Neural Networks performed really well, Accuracy rate of 93% was much higher than accuracy rate of other models used. CNN distinguished between street_music and children_playing class much better than any other model we have used.

Conclusion



Model	Logistic Regression	Decision Tree Classifier	XGBoost Classifier	CNN Neural Networks
Accuracy	79%	**68%**	88%	**93%**
Weighted F1 avg	.79	.68	.88	**.93**

For the purpose of this project we used the most common method available to represent data audio data such as: Mfccs, Tonnetz, chroma etc to represent our data. We extracted data from audio files using librosa. Then we tried to find some insights using 2D and 3D graphs and we ran some hypothesis tests to test the relationship among some features and We created various machine learning models that were trained on train data and tested on test data. Above table shows their performance, We can see that CNN neural networks seemed to have captured our data well and had outperformed other models with and accuracy rate of 93% while labelling audio data in their respective classes.

Future Work

The are many possible directions to take further exploration with this data and learning approaches. We can try to change the way we extract data, we can try to use different methods to extract and represent the data for example: segments were sliced into four second slices. We can try to play around with these numbers. It is important that extracted slice from segment has enough of the original sound to be possible to classify. The process will be difficult and very time consuming. We can also try different neural network architectures trying these might or might not improve classification accuracy without changing our algorithms much.

Recommendations for the client

Automatic urban sound classification can benefit a variety of multimedia applications. In this project, we worked with audio files and identified few problems in the field of automatic urban sound classification: correct method to extract feature, and the difficulty of discriminating sounds in the presence of noise. The automatic classification of urban sounds is relevant in many areas and has a variety of applications including surveillance, highlight extraction, environmental monitoring, video summarization etc. Most Importantly, It also has the potential of improving the quality of life of city dwellers by providing a data-driven understanding of urban sound and noise patterns.