CMSC-6950-001 (Comp Based Tools&Applications 92498)

Project Report on

Urban Sound Classification

Using Statistical Methods for Machine learning

Name: Gubba Dinesh Kumar Guptha

ID: 202195550

Abstract:

Sound Classification is a methodical approach for identifying and analyzing patterns in the Audio. Sound classification can help in many scenarios which are observed in real-world scenarios such as Speech recognition, google smart replies and many more. In this project, we took the Urban Sounds dataset.

Introduction:

This dataset contains audio divided into ten classes, each one representing a different kind of Urban sound, for example, we can find Hammer sound, Children Playing, Car horns, dog barking etc. These classes are taken from the Urban sound taxonomy. And there are 10 folders which contain different audio files with respect to the sound it was actually classified which helps in comparison and reproduction with the automatic classification results.

Downloaded Dataset from :

https://urbansounddataset.weebly.com/download-urbansound8k.html

Just need to fill out the form and download it.

The two main important tasks required to perform are the Conversion of an audio signal to the data representation form and other is processing the data for giving input to divide test and train data.

Dataset Structure:

The Urbansounds8K dataset contains ten folds of audio samples with a length of (<4s).

Slice file name: contains names of audio files.

Fsid : freesound id of the audio file

Class id: As there are ten classes of sounds it represents which class represents the given audio file.

Start: start time of the slice in the original free sound recording

End: represents the end slice in the original free sound recording

Salience: 1- background, 2- foreground

Fold: There are 10 folders in which folder that particular audio file contains.

Class: car_horn,airconditioner,childrenplaying,dogbark,drilling,jackhammer,engineidling, gunshot,streetmusic,siren.

Preprocessing:

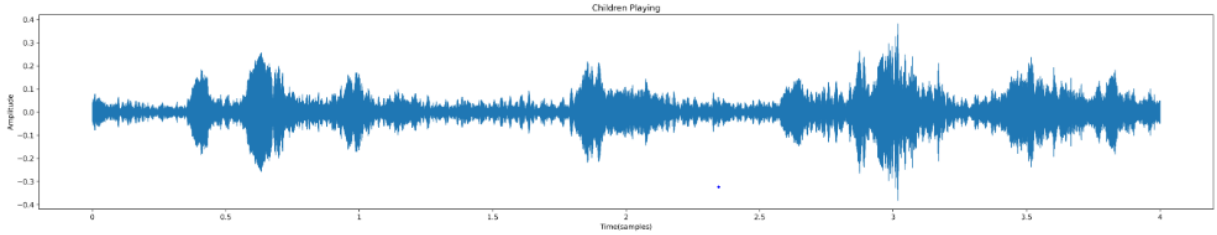
The downloaded data set contains 8732 labelled sound exerts of urban sounds from 10 classes mentioned above. One of the first and main things need to do is to locate the exact path of the audio files.

I wrote one simple function to concatenate the paths and folders and audio file names which are provided in each row of data and created one new column and append it to our existing CSV. Which is named as filelocation contains the exact path of audio located in our local system.

Input:

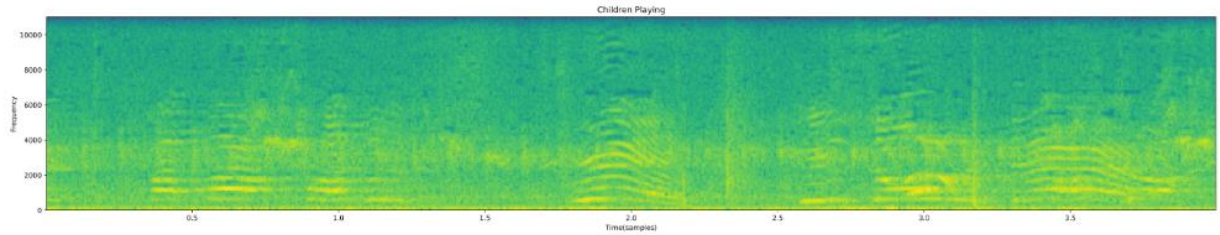
As our main task is to convert the audio file to data representation so whatever the algorithm use we can give them as input. Because we can't give anything in the form of audio to any algorithm directly we needed to process it to the form where it can actually perform some operations and compute the final outcome.

What does the actual signal look like?

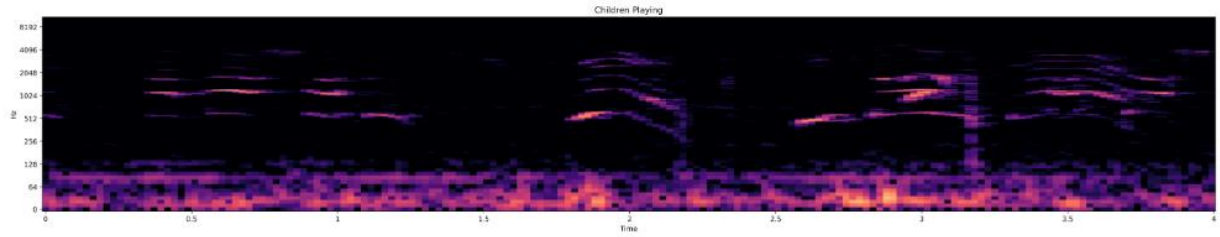


As per the above image you can see that the original soundtrack looks like this and we need to convert this to the form where every algorithm can read them for producing the results.

And to show how the signal looks after performing some major steps like spectrogram and log of the spectrogram. Which actually depicts in what areas the frequencies have been changed and decreased. To save all these data in the form of numerical numbers we need to use some powerful algorithms.

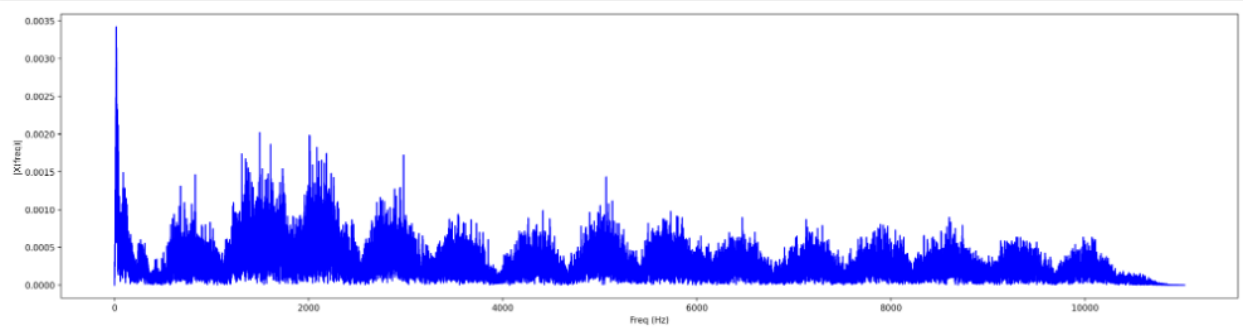


Log power spectrogram



For converting to that numerical data we need to use one of the powerful packages named Librosa. Which has many of the inbuilt methods to analyze the audio file and produces the output in the form of numerical data which is actually having frequencies and some other values which really help in analyzing the data.

The spectrum of a signal which actually represents the sense of when the signal frequencies change and when it fluctuates this is really helpful to clearly understand when the frequency changes.



So, after processing every audio file using librosa we can store the numerical values of the signal and use them for splitting to the train and test data sets. And

finally, X values represent the audio signal frequencies and Y values represent the actual class of that audio file.

Output:

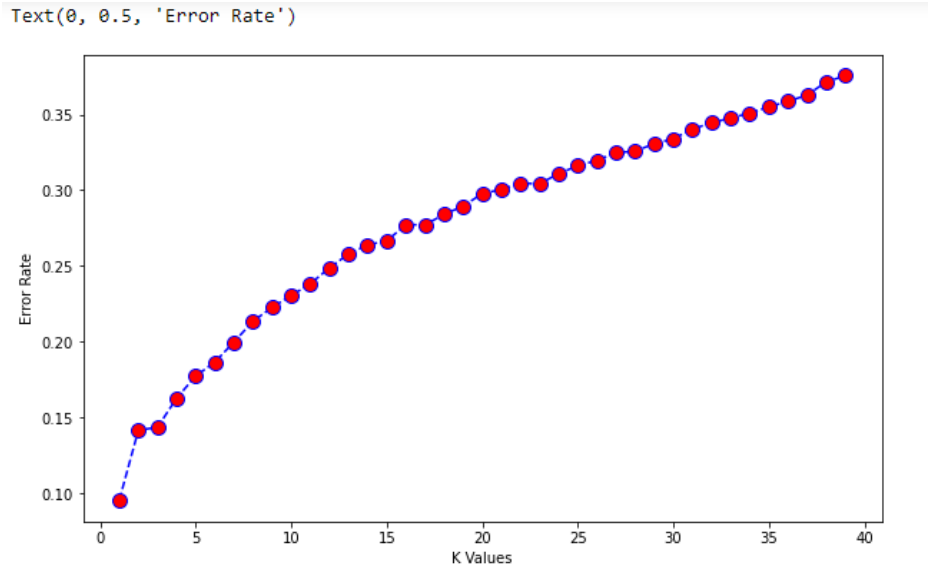
By using the processed test and training data for the Classification algorithms we will be able to know which algorithm is performing well and giving the expected outcomes. In this project, I used four main classification algorithms namely Random forest, Bagging Classifier, Gradient Boosting, and K-nearest neighbours.

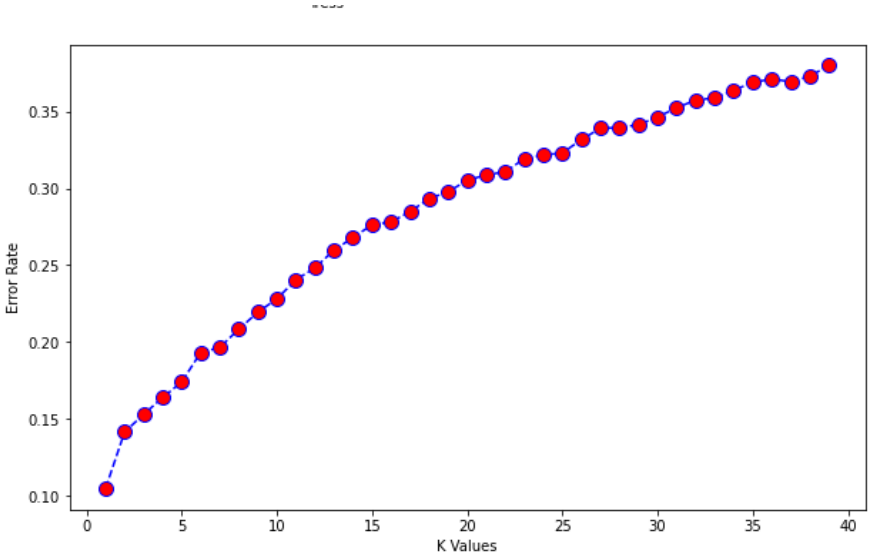
Out of these four, Random forest performed well and gives more accuracy and less error rate when compared to the other ML algorithms we used whereas KNN gave more error rate when the number of neighbours is being increased.

Algorithm	Accuracy
RandomForestClassifier	0.8937242327072835
BaggingClassifier	0.8176820888685296
GradientBoostingClassifier	0.8341731562070546
KNeighborsClassifier	0.8831882730187814

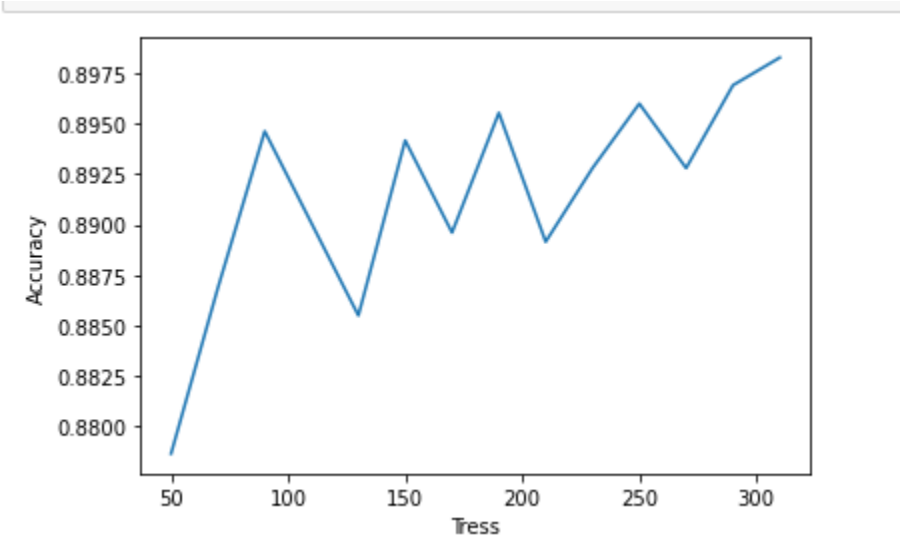
Algorithm	Accuracy
RandomForestClassifier	0.8900595510765003
BaggingClassifier	0.8163078332569857
GradientBoostingClassifier	0.8502061383417315
KNeighborsClassifier	0.9028859367842419

Even though we are seeing the high accuracy in the above picture. The error rate is high when the number of neighbours >=10.





If we closely observe the error rate keeps on increasing upon increasing the K values. So, we won't consider KNN as the best one for classifying the audio files in this scenario.



The above graph is with respect to Random Forest Classification we can clearly see that the accuracy keeps on increasing upon increasing the Number of Tress and getting high accuracy.

Libraries Used:

Some beautiful and powerful machine learning algorithms that we used in this project are really helpful for analyzing and processing the data and producing good results. libraries used as follows,

Sklearn, matplotlib, pretty table, scipy, pandas, NumPy, librosa, SVC, RandomForestClassifier, BaggingClassifier, GradientBoostingClassifier, DecisionTreeClassifier, KNeighborsClassifier , GridSearchCV, StandardScaler, specgram, fftpack, arange, wavfile.

System Specifications:

Operating System: Windows

Softwares: Anaconda, Python, Jupiter

Memory and Ram: 1TB, 8Gb

Conclusion:

By analyzing the above results we can conclude that Random Forest is producing high accuracy and low error rate for classifying the Urban sounds Audio files whereas KNN is producing more Error rate upon the increase of the K-value.

References:

https://www.kaggle.com/datasets/chrisfilo/urbansound8k

https://github.com/tomfran/urban-sound-classification/blob/main/report/report.pdf

https://urbansounddataset.weebly.com/urbansound8k.html