Purav Dipesh Kumar Parekh
Aarti Paan
Sachin A

Music Genre Classification

Finding the best model to classify music.

Introduction

- Machine Learning has several different use cases like classification, regression, prediction on different kinds of data such as images, text, audio, video etc.
- Audio processing is one of the most complex tasks in data science as compared to image processing and other data processing techniques.
- One such application of audio processing is music genre classification. It aims to categorize audio files into specific genres based on various features as we will see further.

What are we doing?

- Classifying audio files/music manually to certain categories/genres is a tedious process. One has to listen to each file for the complete duration & then manually label it individually.
- This process is crucial to determine what makes one song different from another and hence it has to be automated to cut down on manual error and time.
- So, we use machine learning and deep learning algorithms to automate this process.

Why are we doing this? - Motivation

- ➢ We have previously worked with numerical and image data in past projects. We didn’t want to do anything similar again.
- ➢ Our main intention was to ofcourse get a good grade and also learn something new.
- ➢ Hence, we decided to work with audio data.
- ➢ We believed it would help us gain a deeper understanding on how audio files are analyzed and how different it is than numerical and image data.

Dataset

GTZAN Genre Collection (by George Tzanetakis)

Genres: Blues, Classical, Country, Disco, Hip-hop, Jazz, Metal, Pop, Reggae, Rock

	Number of Genres	**Duration**	**Number of Audio files per Genre**	**Format**
GTZAN Dataset	10	30 Seconds	100	22050 Hz Monophonic 16-bit audio files in .wav format

Experimental Setup

1. **Librosa**: It is a Python package designed primarily for music analysis of audio inputs. The building blocks for a MIR (Music Information Retrieval) system are included.
2. **IPython.display.Audio** : It enables you to play audio files directly in the notebook.
3. Scikit Learn
4. Tensorflow
5. Numpy
6. Pandas
7. Matplotlib

What is Audio Data?

Audio data represents analog sounds in a digital form, preserving the main properties of the original. It has three key characteristics to be considered when analyzing audio data:-

- **Time period** - Time period is how long a certain sound lasts or, in other words, how many seconds it takes to complete one cycle of vibrations.
- **Amplitude** - Amplitude is the sound intensity measured in decibels (dB) which we perceive as loudness.
- **Frequency** - Frequency measured in Hertz (Hz) indicates how many sound vibrations happen per second. People interpret frequency as low pitch or high pitch.

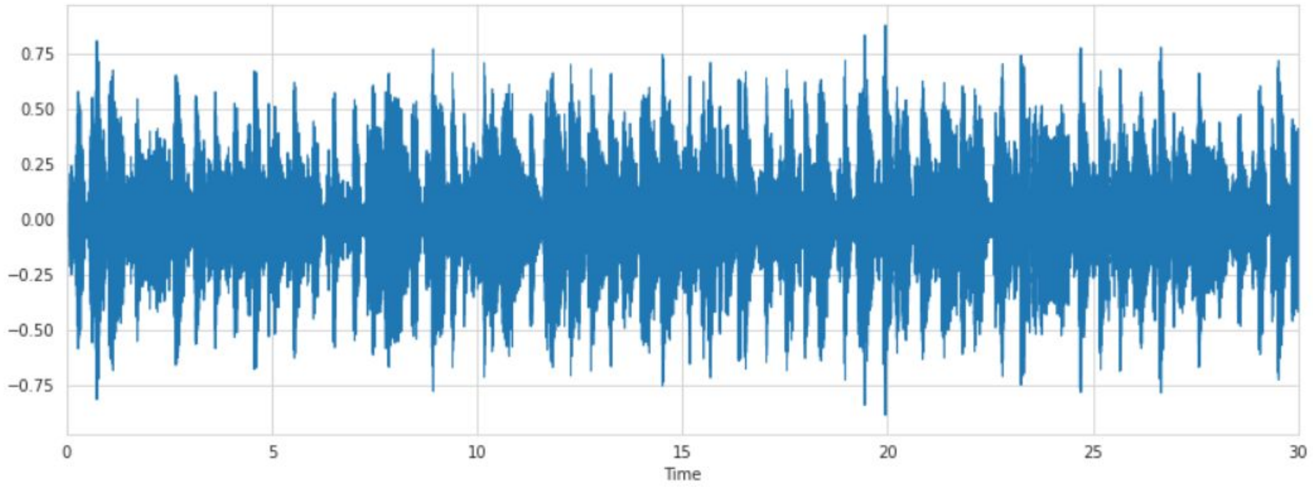
Data Visualization

Data Visualization

- To analyze audio data, there are different features that can be studied and analyzed.
- Some of the features are discussed in the next few slides, which will give us a little insight into the analysis of audio data.

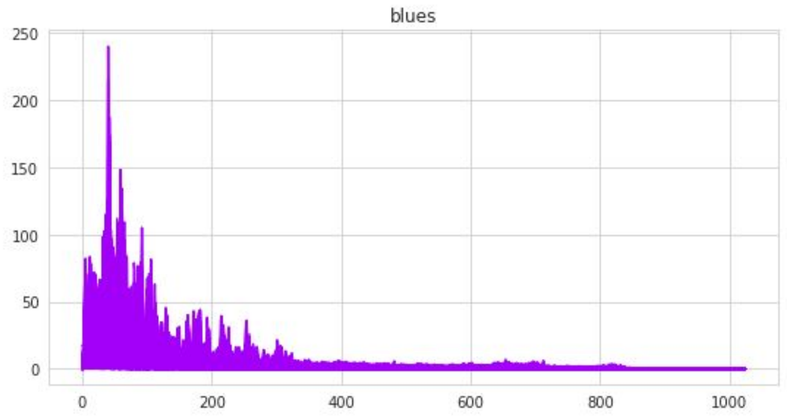
Waveform

A **waveform** is a basic visual representation of an audio signal that reflects how an amplitude changes over time. The graph displays the time on the horizontal (X) axis and the amplitude on the vertical (Y) axis but it doesn't tell us what's happening to frequencies.

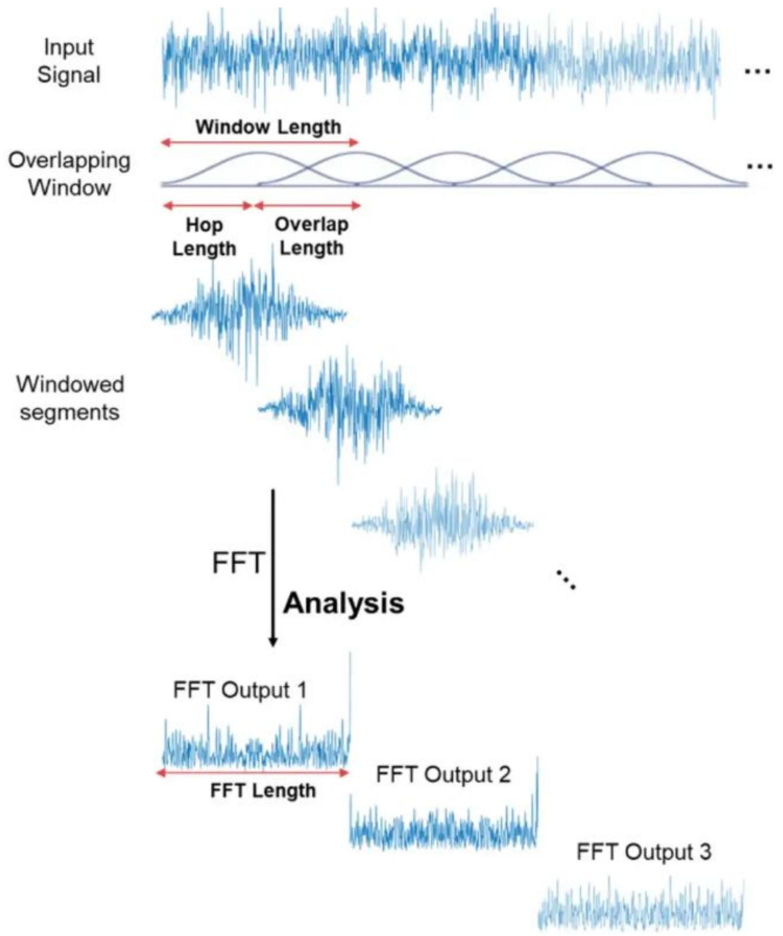


Short-time Fourier Transform

- Fourier Transform is a function that gets a signal in the time domain as input, and outputs its decomposition into frequencies.
- It transforms both the x-axis (frequency) to log scale, and the “color” axis (amplitude) to Decibels, which is approx. the log scale of amplitudes.
- STFT represents a signal in the time-frequency domain by computing discrete Fourier transforms (DFT) over short overlapping windows.



Overview of FFT Process

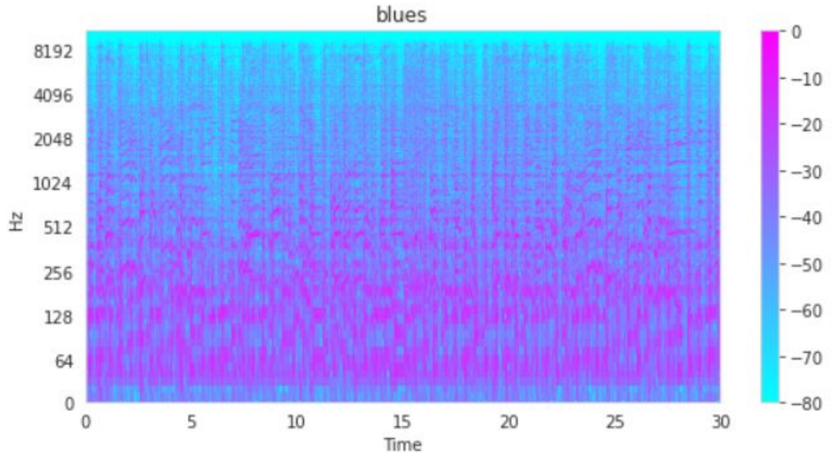


Spectrogram

A **spectrogram** is a detailed view of a signal that covers all three characteristics of sound.

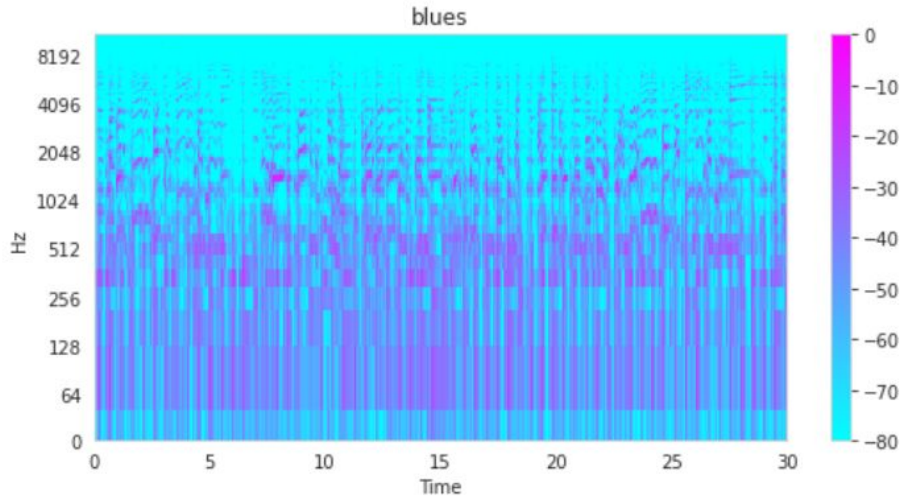
You can learn about time from the x-axis, frequencies from the y-axis, and amplitude from color.

The louder the event the brighter the color, while silence is represented by black.



Mel Spectrogram

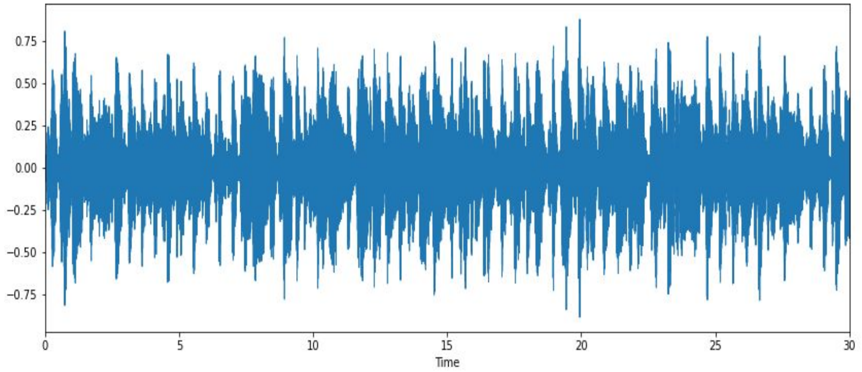
A **mel spectrogram** is a spectrogram where the frequencies are converted to the mel scale.The Mel Scale is a logarithmic transformation of a signal’s frequency.

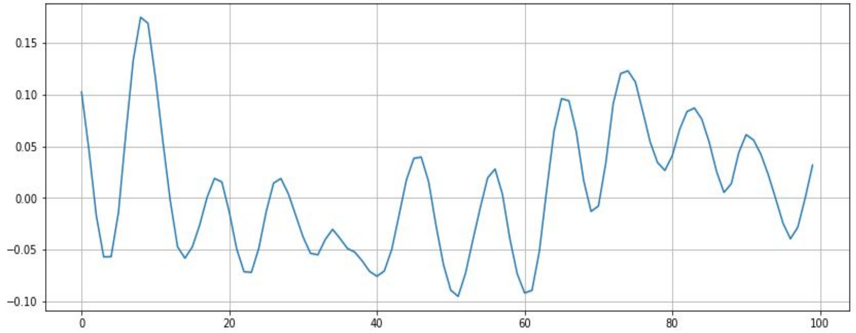


Zero Crossing Rate

The zero-crossing rate is the rate of sign-changes along with a signal, i.e., the rate at which the signal changes from positive to negative or back.

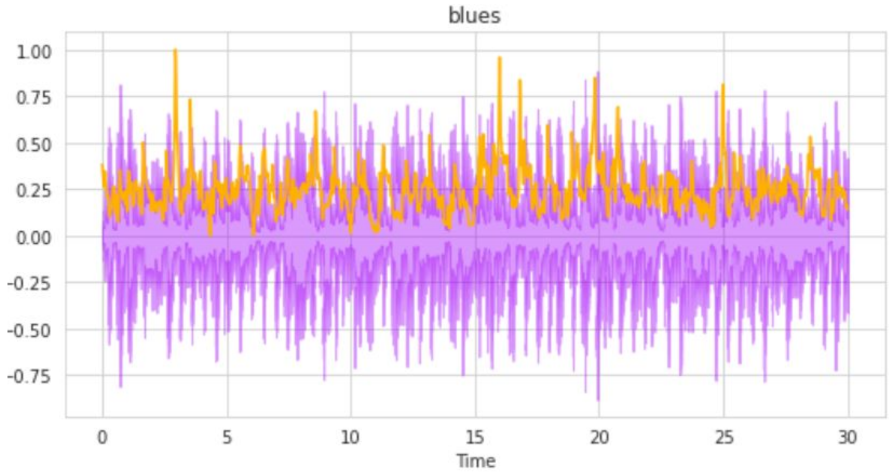
The value of the ZCR tends to be higher for percussively intense music and lower for music that lacks percussive elements.





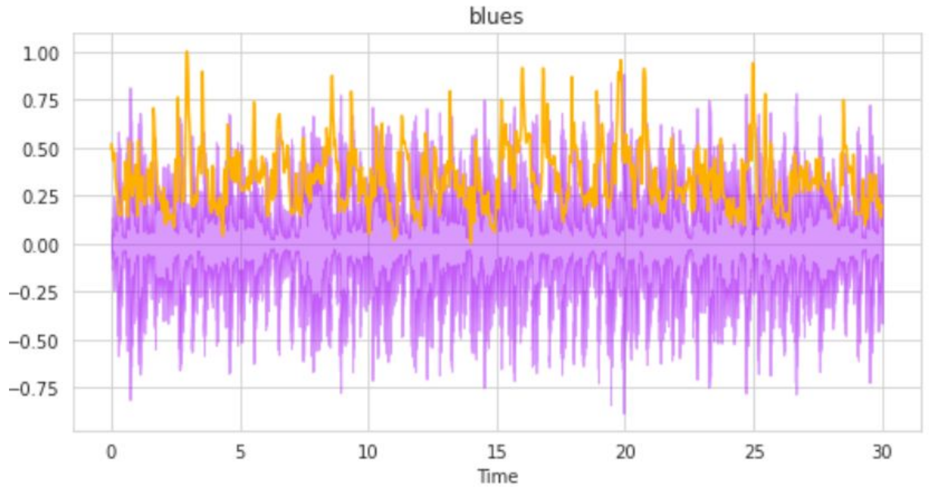
Spectral Centroid

Spectral Centroid indicates where the ”centre of mass” for a sound is located and is calculated as the weighted mean of the frequencies present in the sound.



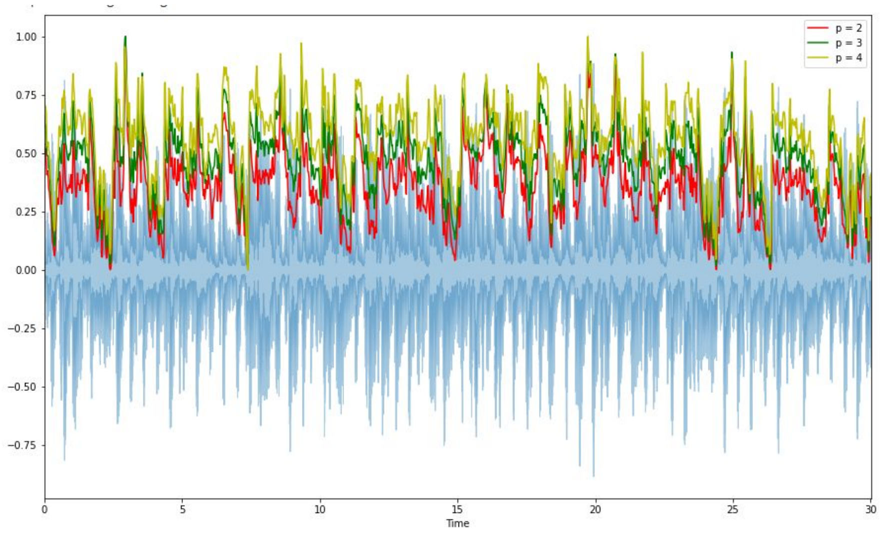
Spectral Rolloff

Spectral Rolloff is a measure of the shape of the signal. It represents the frequency below which a specified percentage of the total spectral energy, e.g. 85%, lies.



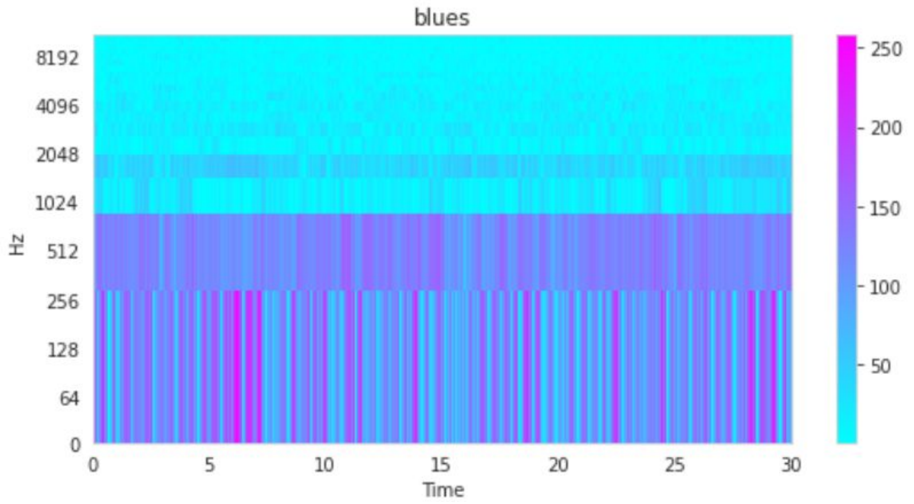
Spectral Bandwidth

Spectral bandwidth measures the difference between the highest and lowest frequencies within a timeframe.



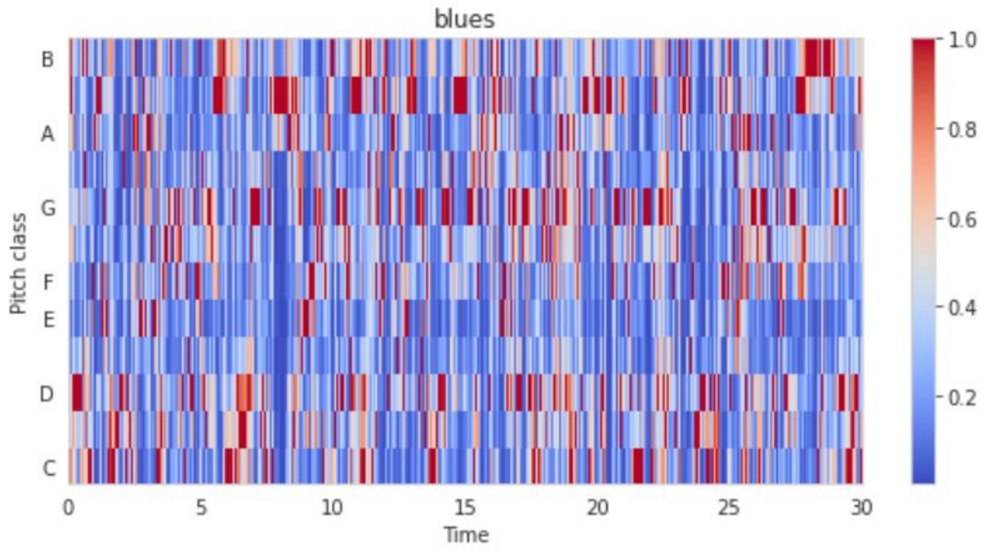
Mel Frequency Cepstral Coefficients

The Mel frequency cepstral coefficients (MFCCs) of a signal are a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope. It models the characteristics of the human voice.



Chromagram

Chromagram sequence of chroma features each expressing how the representation pitch content within the time window is spread over the twelve chroma bands/pitches.



Approaches

Current Approaches

1. Artificial Neural Networks
2. CNNs
3. Other classification models

Our Approaches

1. K-Nearest Neighbor
2. Random Forest Classification
3. Artificial Neural Networks
4. CNN

We started off with KNN as its the simplest classification algorithm and then moved on to Random Forest Classifier.

Next we built our own neural network and after multiple iterations (trial and error) we obtained a neural network that gave sufficient results.

Lastly, we attempted to also build a CNN and use images to make better classification.

Pre-processing

The data we had was in the audio format and we had two options:

1. Convert the audio to numerical data.
2. Convert the audio to image data.

We decided to begin with numerical data.

With the initial 1000 audio files, we built a .csv file that had all the features (rmse, spectral centroid, rolloff etc) in terms of numbers using librosa library.

We also used the encoding technique to transform all the target genres to integers.

Pre-processing

Next step was to standardize/normalize the data.

We used four different methods for normalization

1. Standardization
2. L1 Normalization
3. L2 Normalization
4. MinMax Normalization

Standard Scaling: Standardize features by removing the mean and scaling to unit variance.

L1 Normalization: It changes the data such that the sum of the absolute values remains as 1 in every row.

L2 Normalization: This normalization modifies the data in such a way that the sum of the squares of the data remains as 1 in every row.

MinMax Normalization: The maximum absolute value of each feature is scaled to unit size.

Oops!

Once we had the dataset (~1000 rows and ~60 features) we thought it was ready to be played with.

But we were wrong!

The dataset was too small!

After running KNN, Random Forest and ANN on the data, we never crossed an accuracy of 70%!

We had to find a way to increase the data.

Workaround!

We had to find a way to increase the data!

Alongside the 1000 audio files, we also had a .csv file that contained the same data but only 10 times more.

How!?

Each audio file of 30s, was split into 10 parts of 3s each and its individual feature values were recorded.

Now we had enough data to play with (~10,000 rows)!

Results

BaseLine

The baseline that we have considered is 10% simply because it's the accuracy that we would obtain if an untrained model is given the task of classifying the data.

10% so because we have 10 genres of 100 audio files each totalling 1000 files. Each genre has approximately 10% of the data.

10%

Baseline to beat

KNN

Dataset	**K=3**	**Precision**	**Recall**	**F1-Score**	**Optimal k**
Non-standardized	0.1	0.01	0.1	0.02	Optimal k = 22 0.3063
Standardized	0.59	0.64	0.6	0.58	Optimal k = 3 0.87
L1-norm	0.09	0.01	0.1	0.02	Optimal k = 3 0.89
L2-norm	0.1	0.01	0.1	0.01	Optimal k =3 0.9
MinMax norm	0.91	0.91	0.91	0.91	Optimal k = 3 0.91

Random Forest Classification

Params / Dataset	**Gini Index E=100**	**Entropy E=100**	**Log Loss E=100**	**Gini Index / Optimal E**	**Entropy / Optimal E**	**Log Loss / Optimal E**
Non-standardized	0.8541	0.8651	0.8635	Optimal E = 80 0.8571	Optimal E = 80 **0.8658**	Optimal E = 100 0.8645
Standardized	0.8491	0.8521	0.8588	Optimal E = 100 0.8541	Optimal E = 80 0.8621	Optimal E = 70 0.8578
L1-norm	0.8575	0.8555	0.8548	Optimal E = 90 0.8555	Optimal E = 90 0.8541	Optimal E = 100 0.8498
L2-norm	0.8591	0.8621	0.8591	Optimal E = 90 0.8621	Optimal E = 80 0.8628	Optimal E = 70 0.8611
Max norm	**0.8699**	**0.8705**	**0.8605**	Optimal E = 90 **0.8641**	Optimal E = 70 0.8645	Optimal E = 80 **0.8658**

RF Classification - Performance Metrics for Gini Index and Optimal E

Params / Dataset	**Optimal E**	**Precision**	**Recall**	**F1 Score**
Non-standardized	Optimal E = 80 0.8571	0.85	0.865	**0.85**
Standardized	Optimal E = 100 0.8568	0.86	0.86	0.86
L1-norm	Optimal E = 90 0.8555	0.85	0.86	0.85
L2-norm	Optimal E = 90 0.8621	0.87	0.86	0.86
Max norm	Optimal E = 90 0.8641	0.87	0.87	0.87

Artificial Neural Network

Params / Dataset	Without dropout layers (Train)	Without dropout layers (Test)	With dropout layers (Train)	With dropout layers (Test)
Non-standardized	0.12	0.1024	0.097	0.1185
Standardized	0.9961	0.5505	0.8867	0.5055
L1-norm	0.0998	0.095	0.1047	0.0937
L2-norm	0.8378	0.2395	0.4718	0.2222
Max norm	**0.9983**	**0.8848**	**0.9329**	**0.89389**

KNN vs Random Forest vs Artificial Neural Network

Algorithms / Dataset	**KNN**	**Random Forest**	**ANN**
Non-standardized	0.12	0.8658	0.11
Standardized	0.9961	0.8621	0.5
L1-norm	0.0998	0.8575	0.1
L2-norm	0.8378	0.8628	0.24
Max norm	0.89	0.8699	0.8938

89.38%

Accuracy achieved with Neural Network with dropout layers and the max normalized dataset.

Why does ANN perform better than KNN and Random Forest

1. Even though the results obtained in all the three methods did not vary much, ANN still came out to be a better choice as it performed better by a little over
2. KNN starts to drop in performance when the number of features increase. In our case, we had close to 60 features, and hence KNN didn't perform well as compared to ANN.
3.

Attempt with CNN

After obtaining successful results with numerical data, we decided to analyse the “audio images” using CNN.

We first created a dataset of ~1000 images, each image representing the spectrogram of its respective audio file.

But we ran into issues of overfitting and not enough data. With just 1000 images, we weren’t able to obtain sufficient results.

We tried to find a way to split the audio files into 10 parts of 3s each. After a lot of research and digging we found a lesser known library called pydub that would help us do it with a loop and in an efficient way. But due to the limitation of time we couldn’t proceed further with this approach.

Future Work

Futures & Options

The problem of music genre classification has been researched for long and researchers have made huge strides in finding better algorithms and networks for classification.

This classification problem can also be extended to music recommendation and music matching.

Shazam was one of the earliest apps to successfully commercialise music matching.

Spotify has also tasted huge success with recommending music based on past listening history.

There is a lot of potential associated with audio images that can be leveraged to build better models using CNNs.

Current research has focused more on analyzing spectrograms, but this can also be extended to chormagrams.

Thank You!

Any Questions?

Purav Dipesh Kumar Parekh
Aarti Paan
Sachin A