# Project Report

This project was structured to test the kNN algorithm’s performance on the eclectic work of three modern artists/bands, from completely different genres, which are Jacob Collier, Periphery (Misha Mansoor, Jake Bowen, Mark Holcomb, Spencer Sotelo, Matt Halpern) and Sungazer (Adam Neely, Shawn Crowder). All these artists approach their music by incorporating all sorts of musical elements, each in their own unique way. For each test i gathered three tracks from the latest record releases, and three other iconic tracks, which I subjectively think represent best their discography, being curious if the algorithm would spot some “trademark” of sorts in the selected tracks, or if it would find significant differences, recognizing the eclecticism of these works.

The selected tracks are the following:

+Jacob Collier 🡪 “Ocean Wide, Canyon Deep”, “With The Love In My Heart”, “Everlasting Motion” (from Djesse Vol.1), “Make Me Cry” (single, to be featured in Djesse Vol.2), “Saviour”, “Hideaway” (from In My Room, debut record).

+Peryphery 🡪 “Blood Eagle”, “Garden In The Bones”, “It’s Only Smiles” (from Periphery IV: Hail Stan), “Marigold” (from Periphery III: Select Difficulty), “Scarlet” (from Periphery II: This Time It’s Personal), “The Bad Thing” (from Periphery: Omega).

+Sungazer 🡪 “Bird On The Wing”, “Drunk”, “Electro” (from Sungazer Vol.2), “Dream Of Mahjong”, “Ether”, “I Walk Alone” (from Sungazer Vol.1).

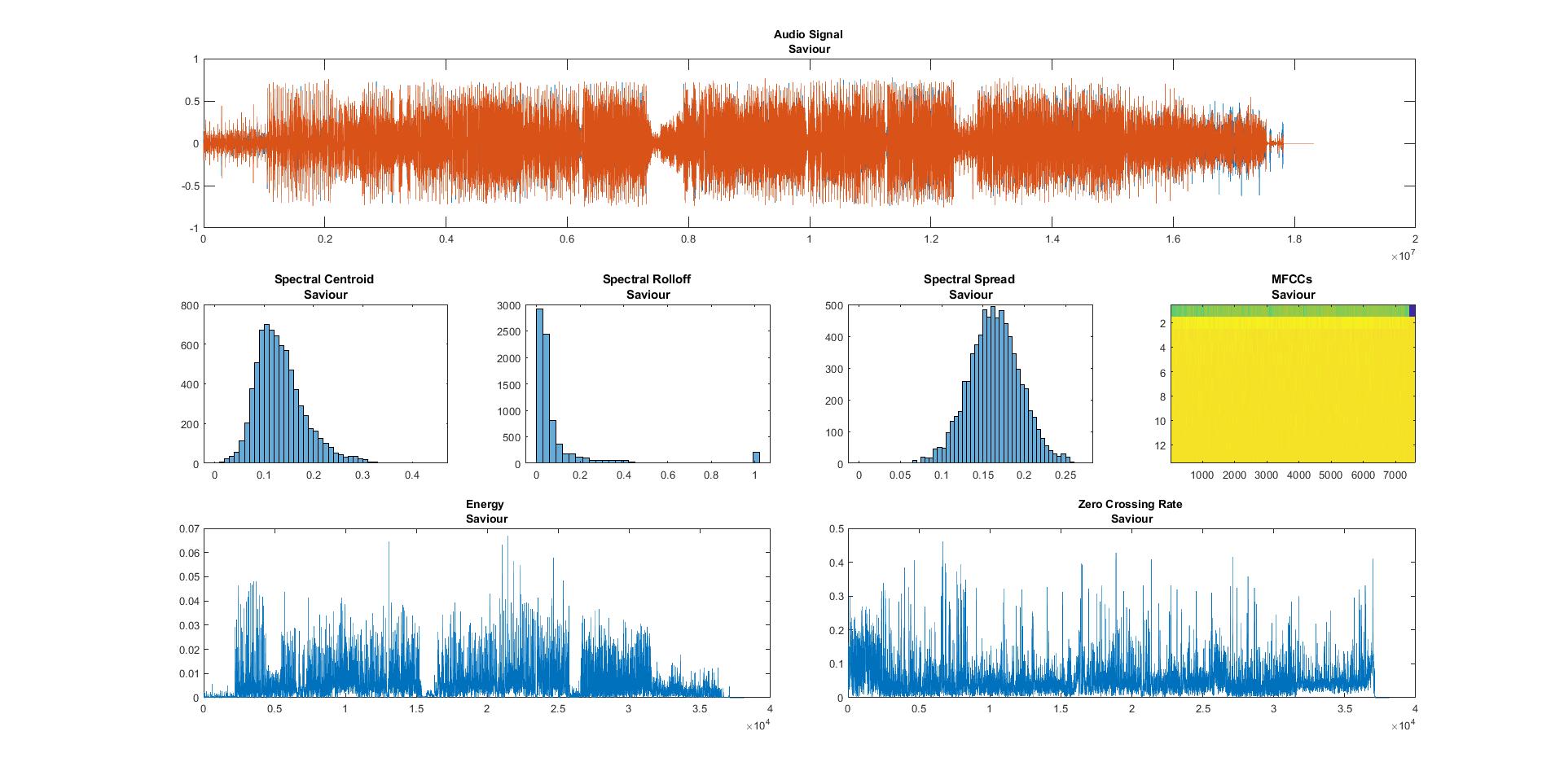
Each artist’s folder is divided in two folders , Original Tests and Processed Silence Tests, one for testing the features of the original tracks, and the other one for testing the features of the tracks whose dynamics below 15.04 dB have been cut out (with Cubase 10, and compressed into a MPEG 1 Layer 3 file), not just to cut out silence, but also to flatten the dynamics, and to see the results of this pre-processing on the kNN’s performance.

In each folder you can find code that extracts frequency and time domain features , plots the audio signal and each feature as in the example below, a test\_feature code (which does a kNN test of the input given features), and 4 test setups, testing every feature. The test1 code puts the first three tracks (those from the latest record) in the train set, and the latter three in the test set, test2 does the exact opposite, cross\_test1 and cross\_test2 crosses the two sets from test1 and test2, swapping one track for set, following my subjective criteria of similarities between the swapped tracks.

Before getting to the commentary part of the report, some bug reports first:

I had some issues with the feature extraction phase, that I tracked down to the windowize function, returning one empty array of windows and a negative number of frames, when given my audio files’ signal in input, yet working perfectly fine with the audio files from the lab. I also tried processing some tracks of my own, recorded and produced on my DAW, in many different formats, with all sorts of sampling frequencies and quantizing bits. In the end the best solution i could get to was removing all the if checks to the number of input arguments, which made everything work well, except for the spectrogram plot, which couldn’t for some reason recognise the audio input signal array, as an array, and threw exceptions all over the place, and that’s the reason for its absence in the plot. I expected the issue to be of the same nature as the windowize function, but I couldn’t manage to do the bugtracking on the built-in functions.

-Example of my\_plot output:



Jacob Collier

For Jacob’s tests i set the window’s length, for the frequency features, to a fith of the number of samples in a second, and the frame size to overlap 25% of the subsequent frames, whereas leaving standard settings (wl = fs\*0.02 and fl = wl/2) for the extraction of the time domain features, the reason for that being his clever and very creative use of dynamics and orchestration to introduce movement and syncopation to the piece, while manouvering the sound scope to a much larger scale, thus i needed less windows on the frequency domain, and a higher number on the time domain. Also i set an array of [1 5 10 50 100 200 500] nearest neighbours, to capture more or less the optimal number, and register in which way the performance decays.

First off, we run the feature\_extract script.

Looking at the plots of the extracted features, frequency wise I’d say my expectations were quite correct, with the centroid, the spread and the MFCCs, often balancing between his bassy voice and the instrumental scape, and the rolloff giving a handy proof of how much he spreads the spectrum with his crazy orchestrations. Also timewise, I expected the data to show how much he plays with the dynamic range, and that was the case.

***I’ll be extensive on my workflow while exposing the tests for Jacob’s original folder, while, to keep it brief, I’ll spend less words commenting the next folders and artists.***

Now, we run the test1 script:

For the spectral centroid we can see a fluctuating recognition rate, maxing out at 500 nearest neighbours, with a 37.27%. For the spectral spread we can see a slowly, but steadily, decreasing performance, with the maximum recognition rate of 32.99% with 1 nearest neighbour. For the spectral rolloff the rate maxes out at 100 nearest neighbours on a 43.02% value, and slowly decays afterwards. Most importantly, for the MFCCs, the results show a 45.27% rate for 500 nearest neighbours, a really good result, considering the stylistic variety of the tracks, and probably mainly due to the presence of his voice and some core instruments in the orchestration. Both the zcr and the Energy features performances max out at 500 nearest neighbors, with 32.68% and 34.1% as rates, slightly more than expected, because of the diversity of the set’s tracks dynamic variations.

Running the test2 script:

The performance on the spectral centroid shows the same fluctuation of test1, but with a higher rate of 42.8% achieved with 10 nearest neighbours. On the spectral spread it shows, instead and opposite tendency, slowly growing to a maximum rate of 42.36% at 500 nearest neighbours, whereas on the rolloff feature the best rate of 44.83% is achieved with 100 nearest neighbours, pretty similar to the test1 results. On the MFCCs it shows a sligthly better performance of 47.11% with 500 nearest neighbours. The performance on the zcr is quite higher with a 43.21% with 500 nearest neighbours, while for the energy it stays around similar values with 32.2% rate, but achieved with 1 nearest neighbour.

Running the cross\_test1 script:

For this test i swapped “With The Love In My Heart” and “Saviour”, between the two sets of test1. On the spectral centroid the kNN performs quite as like as the test1, with a peak of 38.27% with k = 500, while on the spread it slightly outperforms it, with 35.62% at k = 200. On the rolloff it gets, instead, outperformed, achieving just a 39.51% at k = 500, and on the MFCCs too it gets just about 41.54% with k = 500. Zcr and Energy too bring lower performance, with 30.22% at k = 500 and 28.92% at k = 500.

Running the cross\_test2 script:

For this test i swapped “Ocean Wide, Canyon Deep” and “Hideaway”, between the two sets of test2. The overall performances are significantly lower (with a medium decrease ≥ 10%) and not worth reporting, aside from the performance on the zcr, which gets a 42% rate with k = 1.

The overall results actually exceeded my expectations, mostly on the MFCCs tests, where i expected rates lower than 30% at least. It seems that the kNN algorithm, as faulty and primitive as it is, can still catch a hunch of the underlying characteristics that define an artist. As expected, instead, the fluctuation on the results, showed its issues with large pools of data, and the production of sometimes counter-intuitive results (e.g. cross\_test2 poor results or test2 outperforming test1).

**Running the tests on the processed silence files:**

Looking at the test1 results, focusing on the MFCCs (as it’s the most significant, due to the perceptive principles of the Layer 3 compression) for the frequency features, there’s a slight preformance drop, around 1% over the maximum recognition rate, whereas also for the time features, contrary to my expectations, the performance dropped, around 10% for zcr and around 1% for energy. It seems that reducing the dynamic variations has perhaps highlighted the core time domain-wise differences between the tracks. We’ll see how it goes on the following tests.

Test2 results point out a really close performance to the original’s test2 on all the features of interest.

Cross\_test1 results show a better performance on the MFCCs over the original’s, about 5%, and around the same performance on the time features, with slight variations.

Cross\_test2 gives all round much better results than the original’s test, showing a performance raise compared to all the tests done to this point.

Looking at the overall results i’d say the previous assumptions, made commenting test1 results, are mostly wrong, and i could assume that that specific performance drop could be attributed to fluctuations of the algorithm itself, when it comes to big, non normalized, data. I would also say that the dynamic’s cut, and the compression, didn’t really much change the performance, at least on the significant features.

Periphery

The settings for the following tests are: wl = fs \* 0.2, fl = wl /4, both for frequency and time features, and the same array of k numbers as the previous.

Looking at the graphic plots, we can see all the core elements of Periphery’s “big wall of sound”: high loudness, low end tunings, Spencer’s high pitched vocals, and the synths, filling and balancing the overall soundscape. We can see as well some differences, different tunings, or instruments’ range perhaps, or maybe a different mix of vocal techniques (clean/distorted, fried screaming/growl).

**Running the tests for the originals:**

test1 🡪 The overall performances range from >30% to 39%, often best with 10 nearest neighbours.

test2 🡪 There are slight + and – differences with the previous results, also here 10 nearest neighbours is the most recurrent best.

cross\_test1 🡪 for this test i have switched “Blood Eagle” and “The Bad Thing” between sets. Results show again some slight + and – differences, aside from the tests on the MFCCs, which show >40% rates. 1 nearest neighbour seems to be the most recurrent best.

cross\_test2 🡪for this test i have switched “It’s Only Smiles” and “Scarlet” between sets. Overall the best performing test set by far, getting a 45% best rate on MFCCs, and yet again 10 nearest neighbours as the most recurrent best k number.

With large approximation, it’s safe to assume that the results have been stable for the originals’ tests. All these tracks are really different in many subtle ways, so the algorithm generally performed well, picking up a >30% on all features, which i assume comes from the low ends and Spencer’s voice, frequency wise; as for the time features, i expected a bit higher rates on Energy, whereas on the zcr, because it represents, to a certain extent, the noisiness of a signal, it’s a given that it showed the overall best results.

**Running the tests on the processed silence files:**

On all the MFCCs tests we can see a significant increase in the best recognition rates, like 52.1% on cross\_test2 setup, whereas for the time features the performance showed small variations. The most plausible reason for this, is that the pre-processing mostly affected silent sections (as we said, and saw from the graphs, these tracks are generally loud), which are supposedly hard to classify, therefore increasing the performance of the kNN algorithm on the MFCCs. Also the best performance was achieved again with the cross\_test2 setup, which confirms the previous results of the algorith, meaning it probably wasn’t about fluctuations and mistakes.

Sungazer

The settings for the following tests are: wl = fs \* 0.04, fl = wl /4, both for frequency and time features, and the same array of k numbers as the previous. The length of the tracks ranges form 1’ 30’’ to 6’ c.a., every track really different on many levels, and there’s a lot going on in each section, that’s the reason why i wanted more windows to check on.

Looking at the plots, we can see mostly differences between every track. We may expect perhaps lower performances on these tests.

**Running the tests for the originals:**

test1 🡪 frequency features fall under 30%, with 1 as best k number. Results on time features are instead on the same level of average performance we have witnessed so far.

test2 🡪 some variations + and – on different tests, with 1 as the most recurrent k number.

cross\_test1 🡪 for this setup i switched “Bird On The Wing” and “Dream Of Mahjong” between sets. The results for the frequency features are surprisingly high, for example 51.2% on the MFCCs, with 200 nearest neighbours, whereas for the time features the results are on average.

cross\_test2 🡪 for this setup i switched “Drunk” and “Ether” between sets. Generally the performance gets back on average, still being better than the normal tests.

The results met my expectations until cross\_test1, which gave one of the best performances of this project, although it’s probably a coincidence, we’ll see if the last tests will confirm or change the results.

**Running the tests on the processed silence files:**

The overall results confirm the previous tests with a ≤ 5% tolerance. The pre-processing generally lowered the performance on the MFCCs, but not in a significant manner, whereas on time features, results vary in different, but poor ways, on each test. It was interesting noticing that the results on cross\_test1 were still high, meaning that setup switch was a lucky guess.

Drawing conclusions

If for some exceptions, the kNN algorith performed below a 50% recognition rate, and the optimal number of k’s varied between each artist, which proved as a significant result: Jacob’s music gets recognized the most with the most nearest neighbours (500), Periphery’s with a much lower number (10), and Sungazer’s with just 1; my assumption is that Jacob’s music is most suited for this kind of algorithm, as well as Periphery’s , though on a shorter scale, maybe considering just sections between each song, whereas Sungazer’s proves itself as a hard feat for the algorithm on all scales, which also confirms my expectations, as is the most experimental and variating project of the three.