Adverts and Machine Learning

Chris Vanden

January 7, 2018

Abstract

2 machine learning algorithms, SVM and extra trees classifier, were used to classify adverts from 5 different categories in 10 1 versus 1 classifications and a multi-class test. They both gave accuracies of around 80% with some outliers. A visualisation of the data was also made using the t-SNE method.

1 Introduction

After the field of artificial intelligence came to somewhat of a plateau in the 1970's in what is known as the 'AI winter', the field of machine learning arose with its focus being on solving practical problems as opposed to the more grandiose focus of the AI field. It ties in heavily with many area of math involving optimization, statistics and probability theory. The field mainly started to flourish in the 1990's when many algorithms were being made which benefited from large datasets which were now becoming readily available due to the invention of the Internet. The general principal of machine learning is to create a program which can perform accurately on new, unseen data based on the learning of previous data. The introduction of free, open source libraries such as scikit-learn in 2010 and librosa in 2015 has made machine learning much more easily accessible. The advances in deep learning in recent years has made machine learning sophisticated enough to now be used by many large companies in their websites and products such as the images displayed on Yelp or the order of tweets on twitter [3].

What to put in an advert is a huge part of marketing for companies and the choice of the right music and tone of the advert can be crucial in an adverts success. It can be expensive and time consuming to determine what is best for these and so the use of machine learning when choosing an advert can bring many benefits to a company. Adverts have now been around long enough for enough data to have accumulated for machine learning algorithms to make use of and so they can be extremely useful in helping decide on the best advert choice for a company.

2 Theory

2.1 Fourier Transform

Audio signals are continuous in time and can be represented as a discrete time series which represents the signal by a series of values of the amplitude for each frame. The number of frames is determined by the sample rate which is the number of times the audio is sampled per second. The Nyquist–Shannon sampling theorem dictates the sample rate to be used in order to lose no information of the original signal [4]. It shows that the sample rate required must be at least twice that of the highest frequency present in a sample, known as the Nyquist rate. If for a given sample rate there are frequencies present above half its value, aliasing will occur causing distortion as these frequencies will be determined incorrectly. A Fourier spectrum can be calculated from the time series to represent the frequency and phase information as it changes over time. There are different methods of calculating the spectrum, the one used for this project is known as the short-time Fourier transform (STFT). This can be done either continuously or discretely, the discrete-time method is done for this project. This method involves splitting the time series into small sections known as windows and calculating the Fourier transform for each of these windows individually. The windows are spaced out by a number of frames known as the hop length [5]. The hop length is chosen to be smaller than the window length so that the windows overlap, allowing for a potentially smoother spectrum to be created. The calculation of the discrete-time STFT can be represented as:

$$X(m,\omega) = \sum_{n=-\infty}^{n=\infty} x(n)\omega(n-m)e^{-in\omega},$$

where $x(n)$ is the value of the time series at time n and $\omega(n-m)$ is the window function defined by m. Direct computation of the STFT is relatively slow and so the fast Fourier transform (FFT) is generally used to calculate it. It makes the STFT faster by factorising the frequencies into evenly sized bins [6]. The resultant speed of the transform is dependent on the number of bins, N. One main aspect of an STFT is the time resolution versus frequency resolution. Having a high time resolution gives the precise times when the frequency changes but makes it hard to see the exact frequencies. A high frequency resolution makes it easy to see the specific frequencies but not the times when it changes [7]. A spectrogram can be obtained from the squared magnitude of the STFT, an example of which is shown in Figure 1.

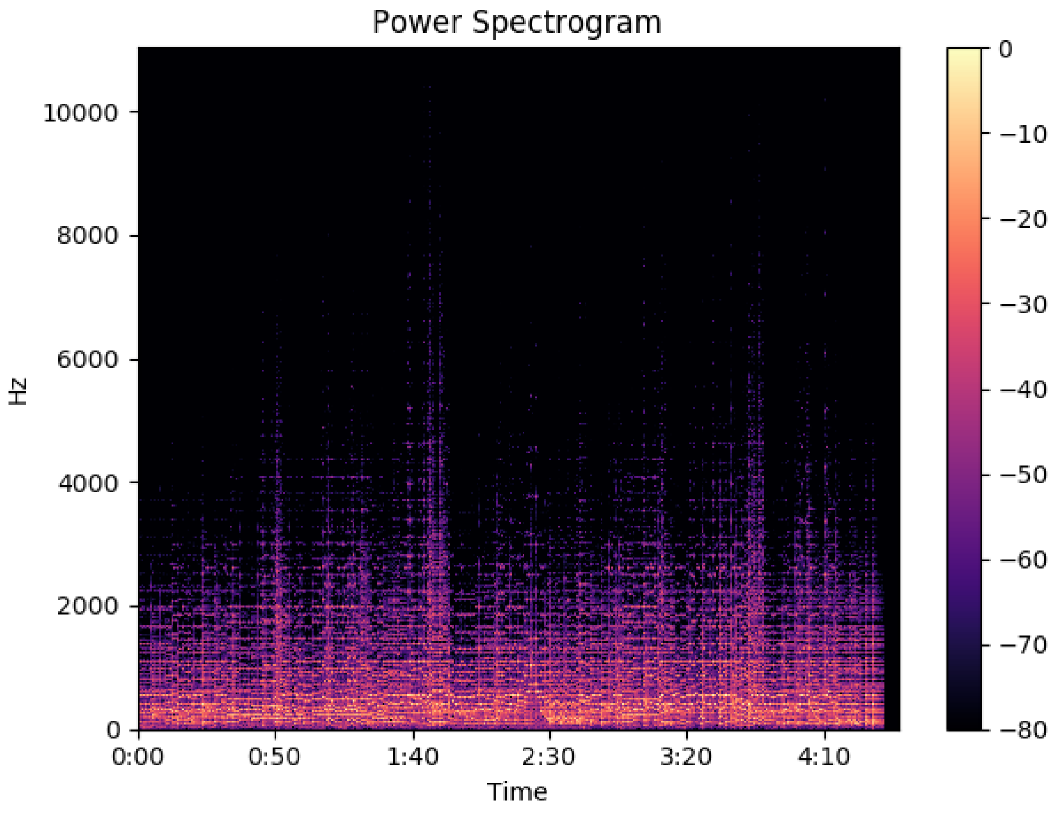

Figure 1: Example power spectrogram made from the squared magnitude of the short time Fourier transform of the song Moonlight Sonata. The brightness of the frequencies represents their corresponding energy.

2.2 Audio Features

Most of the features are calculated within small frame windows which are translated across the spectrogram. This allows the features to be calculated locally, showing how the values change throughout the sample. Some of the features are calculated directly from the time series as they do not need the spectral information such as the zero-crossing-rate. This is a measure of the rate at which the signal changes between positive and negative. Its value is determined by which frequency is most present and is generally determined by the lowest frequencies of a signal as when higher frequencies are convolved with lower ones, the lower frequency will be the determinant how often the signal crosses zero [8]. As a result this feature is important in determining if speech is present in a sound sample as vocal frequencies are very low. The other non-spectral feature used is the tempo which simply measures how many beats are present throughout by using the onset strength which looks for peaks in the time series, an example onset strength plot is shown in Figure 2 [9].

The rest of the features use the spectral information. Spectral centroid: This calculates a weighted mean of the spectrogram for each frame by treating each frame as a normalised distribution where the amplitudes of the frequencies correspond to their weights [10]. Chromagram: A chromagram splits a sample into pitch classes. Pitch classes contain all pitches of an integer number of octaves apart. The most common tuning system used is the twelve-tone equal temperament which separates each octave into 12 semitones, [C, C♯, D, D♯, E ,F, F♯, G, G♯, A, A♯, B] [1]. An example chromagram is shown in Figure 2. RMSE: Calculates how loud the signal is by getting the total magnitude for each frame window and calculating the root-mean-square energy. If all samples used are normalised, this tells the relative loudness of each sample. Spectral Bandwidth: Calculates the frequency difference between the lowest and highest frequencies present at each window. Spectral rolloff: This is the frequency under which a defined percentage of the spectrum is contained. The percentage is generally taken to be 85% [11]. Poly features: Fits an n^{th} order polynomial to the frequencies for each frame. A 2^{nd} order polynomial was chosen for this project returning 3 values: the quadratic, linear and constant coefficients. Tonnetz: Computes the tonal centroid features using the method from [2]. This is done by converting the 12 chromagram feature vectors from before into a 6 dimensional vector which can be represented with 3 axes: fifths, minor thirds and major thirds.

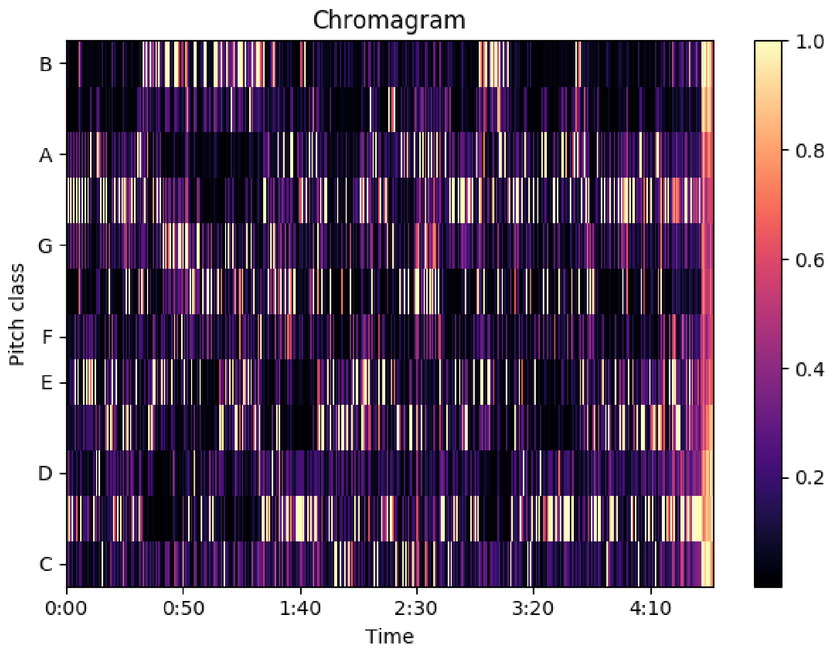

Figure 2: An example chromagram for the song Moonlight Sonata. The brightness of the vertical bars represent the amount of that pitch class present at that time int he song.

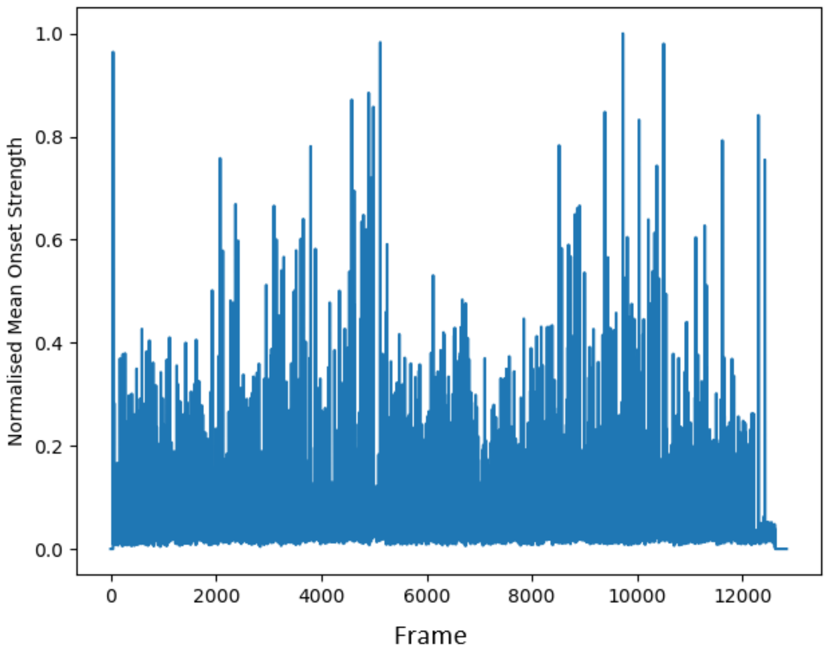

Figure 3: Example onset strength plot for the song Moonlight Sonata. The peaks in the graph correspond to the time of the beats in the song.

2.3 Extra Trees Classifier

The random forest classifier is a supervised learning algorithm used to predict which class something belongs to based on boundary conditions made from training data and averaging the predictions given by the ensemble of different random trees. The classification itself is based on a decision tree model. A decision tree is made up of nodes and leaves. The nodes each contain a feature which is used to split the data at that point. The leaves contain class labels given to any data points ending up there. The decision tree works by taking in all of the data and analysing every feature for the best split at each node. The tree ends when either all points reaching a node are of the same label or are at the minimum number of points required to end the algorithm, as specified by the trees parameters. The number of trees used increases the accuracy of the ensemble averaging up until a point of convergence, above which increasing the number of trees increases processing time for no significant gain [12]. Decisions trees work on a top-down basis and generally use what is known as a 'greedy' algorithm when determining the best split. This looks for whatever split best labels the data correctly at the current node without considering the effect this will have on further nodes down the tree. This can lead to sub-optimal splits being chosen and so there are different metrics which can be used to determine a better split by considering the sub-sets of data as well as the current node. The metric used for this project is called the Gini impurity. This looks to minimise misclassification by calculating how often a data point would be mislabeled if it was randomly given a label of the available choices of labels in the subsets, J [13]. The equation for calculating the Gini impurity is

$$I_G(p) = 1 - \sum_{i=1}^{J} p_i^2,$$

where p_i is the fraction of data points labeled i in the subset. This will have a maximum value of $1 - 1/J$ when there are equal amounts of each class in the subsets and a minimum value of 0 when all of the data in the subsets belong to the same class. Using this metric along with the decision tree is known as the CART algorithm. One of the disadvantages of decision trees is they are very susceptible to over-fitting the data, giving too much value to potential misleading noise in the data set. This results in them having a low bias but high variance and the extra tree classifier adds various steps of randomness to account for this in a bias-variance trade off [14]. The bias in a classifier is a measure of how much it under-fits the data, meaning how much of the important information between the features and class labels is not being identified due to the methods used in the classifier. The first attempt to reduce the variance in the CART algorithm was the use of bootstrap aggregating, also known as tree bagging. Instead of using all of the data in every tree, this method instead assigns each tree a random sample with replacement of the original data [15]. Each tree will now only have a fraction of the initial data sample, with the potential for a tree to contain the same data point multiple times due to the replacement. This variation of the algorithm has no added parameters as the randomisation is done implicitly, based on the total number of samples, N. This randomness reduces the variance which can come from some of the samples, in this case adverts, containing outlying features which would be heavily weighted by the node splitting. Bagging can potentially increase the bias of the classifier however as some important data may be overlooked due to exclusion in the random samples. The random forest method takes the randomisation a step further by only allowing each node to search a subset of the features when looking for the best split, further decreasing the variance while increasing the bias. This method has been shown to be more accurate than both the CART and bagging methods. A parameter is introduced in this method of the number of features to consider at each node. This generally works well when set to be $\sqrt{n}$, which is how it was set in this project. The extra trees classifier takes the final step in adding randomness by generating a random split for each of the features searched at each node and returning the best of the splits generated [16]. This method of randomising the splits has been shown to be very effective at reducing the overall variance of the classifier.

2.4 Support Vector Machines

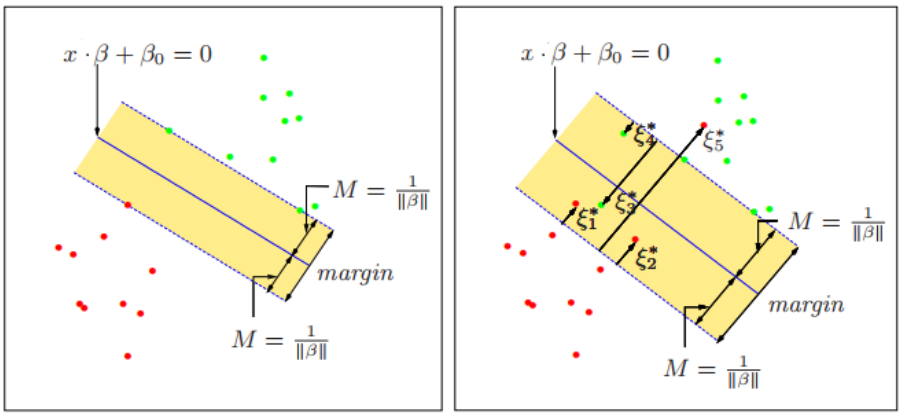
$x \cdot \beta + \beta_0 = 0$
$M = \frac{1}{\|\beta\|}$
margin
$M = \frac{1}{\|\beta\|}$
$x \cdot \beta + \beta_0 = 0$
ξ_4^*
ξ_5^*
ξ_3^*
ξ_1^*
ξ_2^*
$M = \frac{1}{\|\beta\|}$
margin
$M = \frac{1}{\|\beta\|}$

Figure 4: The plot on the left shows data which is completely linearly separable resulting in a hard margin. The plot on the right shows data which cannot be fully linearly separated and so has added slack variables to allow points to cross the margin.

Support vector machines are supervised learning algorithms that work in a similar way to extra tree classifiers by training on data to then attempt to classify new data based on the resultant boundaries created. The difference is in the algorithm used to train and classify the data. SVMs, unlike random forests, are not naturally multi-class classifiers as the algorithms used are based on finding the best fit of 2 classes. There are methods that can be used to convert SVMs to work on multi-class problems which will be discussed later. The objective of an SVM is to create a hyperplane that most accurately separates the 2 classes while maximising the distance between the plane and the nearest points of each class. The sample data is taken in the form $(x_1, y_2), (x_2, y_2), ..., (x_n, y_n)$ where n is the total number of samples and x_i is a real vector with dimensions equal to the total number of features. y_i are the class labels and are normally set to be -1 and 1 for the 2 respective classes. The points are plotted in what is known as feature space and the points which lie closest to the plane are known as support vectors and they are the points which define the final placement of the plane as these points define the region most subject to noise. The hyperplane is defined to be

$$\boldsymbol{\beta} \cdot \boldsymbol{x} - \beta_0 = 0$$

where $\boldsymbol{\beta}$ is the unit vector normal to the hyperplane and β_0 is the hyperplanes offset from the origin along $\boldsymbol{\beta}$. The final hyperplane is made by first creating 2 separate hyperplanes which both separate the classes and then maximising the distance between them. The distance between them is equal to $2/||\boldsymbol{\beta}||$ which defines the region known as the margin and the final hyperplane is placed at the centre of it. In simpler cases where the 2 classes are completely linearly separable, the 2 hyperplanes can be taken as 'hard margins' which means they are defined so that no point can be on the wrong side of the margin. To maximise the distance between the hyperplanes, $|\boldsymbol{\beta}|$ must be minimised. Putting this together leads to the optimization problem:

$$minimise \; ||\boldsymbol{\beta}|| \; subject \; to \; y_i(\boldsymbol{\beta} \cdot \boldsymbol{x}_i - \beta_0) \geq 1, \; i = 1, ..., n$$

For cases where the data is not linearly separable, slack variables are introduced to allow for points to cross over onto the wrong side of the margin. They can be defined as $\xi = (\xi_1, \xi_2, ...\xi_N)$ where ξ is a measure of the distance in which a point overlaps the margin. The constraint in equation X can now be written as

$$y_i(\boldsymbol{\beta} \cdot \boldsymbol{x_i} + \beta_0) \geq M(1 - \xi_i),$$

where $M = 1/||\boldsymbol{\beta}||$ [14]. Here the overlap is measured in relative distance to the size of the margin rather than the pure distance. This allows the solution to be more efficiently obtained and is the formation generally used for most algorithms. The algorithm used for this project is called SVC and uses the python library *libsvm* to solve the optimization problem.

Multi-class problems are treatable by using either a 1 versus 1 or 1 versus rest approach. The 1 versus 1 approach works by treating each pairing of classes separately and calculates a different result for each one, resulting in $n(n-1)/2$ classifiers being made. The 1 versus rest approach takes a single class to be positive (+1) and the rest of the classes together as negative (-1), resulting in n classifiers being made. When predicting labels for test data, this approach gives the result of whichever classifier has the highest confidence score which is a measure of the distance from the samples to the hyperplane. Sometimes a linear separation isn't the best way to separate the data and in these cases a 'kernel trick' is used to transform the feature space into a higher dimensional space where the points can be more easily linearly separated. This is done by replacing the dot product kernel originally used with a non-linear kernel. Once transformed, the algorithm works in the same way as before at the result is then transformed back into the original feature space.

2.5 t-SNE

The t-distributed stochastic neighbor embedding technique is a way to transform high dimensional data sets into low dimensions, typically 2 for visualisation. It is a variation of the SNE method which works by assigning the euclidean distance between pairs of data points as probabilities , p, which represent the similarity between them. The probabilities are assigned based on how likely the point x_J is to be the neighbour of x_I if assigned in proportion to a Gaussian density distribution plotted at x_I. Since only pairs of points are being considered, the probability $p_{I|I}$ is set to 0. Similarly, points on the reduced data set can be given probabilities in the same way, labeled q. The measure of how well the method has reduced the data is then a measure of how well the values of $q_{I|J}$ match their $p_{I|J}$ counterparts. The difference between the 2 is called the Kullback-Leibler divergence and minimising this is the goal of the method. This cost function, C, is given by

$$C = \sum_I \sum_J log(\frac{p_{J|I}}{q_{J|I}})$$

and is solved using a gradient decent method. One of the problems the SNE method has is that different types of errors contributing to the cost function are not weighted equally. For example there is a large cost in using a small q value to represent a large p value but only a small cost for the opposite error. The t-distributed method works to improve this by replacing $p_{J|I}$ with p_{JI} and $q_{J|I}$ with q_{JI} in order to make the probabilities symmetric. The other problem is that the minimisation of the Kullback-Leibler divergence is difficult to optimize and suffers from something known as the "crowding problem" which relates to the difficulty in accurately modeling small distances on the transformed dataset. The t-distributed method improves this by using a student t-distribution for modeling the mapped dataset as opposed to a Gaussian [17].

3 Method

3.1 Getting Data

The platform used to collect the data was youtube. For initial testing purposes, 30 rap and 30 classical songs were downloaded. These 2 genres were chosen specifically as they seem like they would have significant differences between them, allowing them to be easily separated which would give better insight to whether the classifiers were working correctly. It was useful to have a small data set to use for testing as the time taken to load the songs to took a long time when done on large data sets. This allowed for multiple iterations to be done on the code without having to wait on the large runtime in between. Since these songs were only used for the purpose of testing, the small amount of data from them was not enough to get any meaningful results.

In order to get an adequate amount of data for the main testing of the classifiers with adverts, a web scraper was used to search youtube. This allowed for a large amount of data to be collected in a much shorter time period than if it were to be done manually. The web scraper works by querying youtube with a specified search and downloaded every result for a specified number of pages. Initially, only 2 types of adverts were downloaded, car and perfume, to test binary classification. This was later expanded to 5 classes with the addition of toy, cosmetic and christmas adverts. It was found that the results for most search queries were only relevant for the first few pages and so to account for this the search queries for advert categories were split up into different brands belonging to that type of advert. The search queries used were:

- Perfume: Chanel, Dior, Hugo Boss, Calvin Klein, Giorgio Armani

- Car: Nissan, Toyota, Audi, Bmw, Volvo
- Toy: Hasbro, Fisher Price, Mattel
- Cosmetic: Nivea, Tresemme, Garnier, Loreal, Head and Shoulders
- Christmas

The searches were made with 'tv advert' atatched to the end of the queries in order to make the results more relevant, reducing the occurrences of things such as documentaries about the brand. Christmas was not split up into different brands since many brands make christmas adverts as well as their normal type of advert. As a result, only 120 christmas adverts were obtained compared to the 400 obtained for each of the other 4 classes. While some of the videos taken still included things such as advert compilations which would skew the data, having a large amount of data worked to minimise the effects of these by making them only a small portion of the dataset.

3.2 Feature Extraction

The python library librosa was used to extract the audio features from the samples. The first step in the extraction process involved loading in the songs at a specified sample rate. Since most adverts will not contain frequencies above 10000 Hz, a sample rate of 22050 Hz was chosen based on the Nyguist theory mentioned in section 2.1. This sample rate was high enough to not incur problems with aliasing. The load command in librosa stores the data as a 1 dimensional array with each element corresponding to each frame of the song. Since loading in the data was a time consuming process, the arrays were saved so that they could be easily loaded in for the next steps of the feature extraction without having to re-sample the data every time. It was found that normalising the data by dividing each sample by its highest value was needed to make the energy features relevant as without this normalisation, the volume set for the individual adverts on youtube would prevent them from being comparable.

Once the data had been loaded and stored in arrays, the next step was to calculate the STFT and create power spectrograms for each sample. The number of frequency bins was set to 2204 Hz in order to make the frequency bins 10 Hz in size which was deemed to give an adequate frequency resolution without being too time consuming. The resultant power spectrograms were also split up into their harmonic and percussive components as these individual components could give better results when used for some of the features as opposed to the overall spectrogram. In order for the features to be usable with the machine learning algorithms, they had to each be made into single data points. Most of the features get extracted as an array of the features values throughout the sample and so the mean was calculated for each feature to give single values to represent them. Since the variation of the features throughout the samples can also contain useful identifying information, the standard deviation of each feature was also calculated. The delta values were also calculated for each feature which give their first order time derivative and the mean and standard deviation were also calculated for these. For getting the energy features, the spectrograms were split up into different frequency ranges in order to get the energy for different types of frequencies. The frequency ranges chosen were 0-80 Hz, 80-320 Hz, 320-1280 Hz, 1280-5120 Hz and everything above 5120 Hz. In music terms, these ranges correspond to bass, mid bass, middle, low treble and high treble respectively. The RMSE was calculated for each of these ranges using the overall power spectrum as well as the individual harmonic and percussive components. For the chromagram, the mean and standard deviation were taken for each of the 12 pitch classes and similarly for the different axes of the tonnetz. The rest of the features could be calculated directly from the spectrums and audio time series without any extra steps. The overall number of features for each sample was X. It was found that the time taken to extract features from the samples did not increase linearly with duration but instead went from taking a few seconds on shorter samples (a few minutes long) to over 10 minutes on longer samples (over 5 minutes long). To account for this, any samples which had a duration of over 5 minutes were deleted before starting the extraction process. This also helped to get rid of samples which were likely not actual adverts as adverts are generally never longer than a few minutes. It was also found that some of the samples contained invalid data as the videos had been muted on youtube. These samples were identified by checking if any of the data points contained 'nan' values as these would be present if the max values divided by in the normalisation were 0. Multiprocessing was used to vastly speed up the process of both loading and extracting features by allowing multiple samples to be done at once. It was set up so that it could work on 16 samples at once, essentially making the process 16 times as fast as without it.

3.3 Machine Learning

To get an idea of how similar the different advert classes were to each other, a similarity matrix was made by first calculating the average value of each feature for each of the classes. The pairwise euclidean distances between the classes was then calculated and converted to a square distance matrix. This could then be plotted to show which classes had large distances in their features overall and which were closely related. For clustering visualisation the t-SNE algorithm was used. Both this clustering algorithm and the classification algorithms used come from the scikit-learn python library. The dimensions of the embedded space was set to 2 so that a visual representation could be plotted. There are 2 main variables, perplexity and learning rate, which affect how well the t-SNE collapses the data and it was found that these required careful tuning in order to produce a good visualisation. Another variable which can have an effect on the result is the maximum number of iterations the algorithm does before finishing however as long as this is set high enough to where the algorithm stops before reaching it, it won't affect the result. To find the optimal value for the perplexity and learning rate, one was held constant while plots were made for varying values of the other. Having a learning rate too low resulted in the data being clustered in a tight ball whereas having it too high resulted in the points all being separated at roughly equal distance from each other. Perplexity generally gives a better result the higher it is however after a certain point, normally once it becomes greater than the number of data points, it can start to lose the clustering of the data. The values chosen which gave the best clustering of the classes were X and Y for the perplexity and learning rate respectively.

The data was split up into data to use for training and data to use for testing. The higher the percentage used for training, the better the algorithms performed as having more data to train on allowed them to generate better classifications however it also meant the deviation in the results became higher as the number of test points decreased. A split of 80% training data was found to give the classifiers the best scores without making the standard deviation too high. Since the classification algorithms use supervised learning, class labels needed to be provided along with the data. The labels used for the classes were '0' and '1' for binary classifications and 0 through 5 for multi-class. In order for the scoring of the classifiers to be done correctly, the class with the lowest number of data points had to be labeled '0' as this gets treated as the positive class for the purpose of type I and II errors. The first classifier tested was the extra tree classifier. The number of trees used was 250 while the rest of the parameters were left to their default values. Increasing the number of trees up until around this value came with a significant increase in the score of the classifier however increasing it past this point had little to no effect on the score while increasing the processing time. For scoring the classifier, 100 repeats were done. The data was shuffled for each repeat meaning the algorithm used different data for training and testing each time. This allowed for a more robust score to be calculated, reducing the bias of training and testing on the same data.

The score for an algorithm is a measure of the percentage of classes it correctly labels. Metrics of recall, precision and f1 were also taken. These give different ways of measuring how well a classifier has performed based on true and false positives and negatives. The types of labels of the classes as well as the different types of error are shown in Table 1.

	Positive	Negative
Positive	TP	FP (Type I error)
Negative	FN (Type II error)	TN

Table 1: Table showing the different types of labels a prediction can have of true and false positives and negatives as well as the 2 types of error. The columns represent the predicted values while the rows represent the true values.

The equations for the metrics are:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall \; = \; \frac{TP}{TP + FN}$$

$$F1 \; = \; 2 \times \frac{recall * precision}{recall + precision}$$

Precision is a measure of how many of how well the classifier is at not labeling negative samples as positive. Recall is a measure how well the classifier labels the positive samples. F1 is a weighted mean of the precision and recall values. The classifier was tested with 10 binary classifications of all of the combinations of classes as well as a multi-class test using all 5 of the classes. A normalised confusion matrix was created for the multi-class test, giving the percentage that each class was labeled as. As well as getting scores for the classifier for these tests, an importance ranking was also obtained from the multi-class test. Getting rid of the lowest ranking features does not improve the accuracy of the extra trees classifier since low ranking features are already weighted very little and removing a feature will only ever reduce the accuracy as the classifier has less information to work with. The feature ranking is still useful for gaining insight into which features are the most identifying when it comes to distinguishing classes.

The next classifier tested was the support vector machine. When first trying the SVC it was found to run extremely slowly. This was fixed by normalising the features by dividing each feature by the highest value of that feature from all of the classes. When trying to run the classifier with all of the data it slowed down dramatically, even with the features normalised. To account for this, only the best 40 features were used, as obtained from the importance ranking from the extra trees classifier. The class weight parameter was set to 'balanced' meaning more weight was given to the lower data classes such as the christmas class. When testing the kernel types, it was found that the linear kernel gave the best results. The polynomial and radial basis function kernels performed very poorly and so were not used. The SVC was tested in the same way as the extra tree classifier; 10 binary classifications as well as a multi-class test. The one versus rest mode was used when doing the multi-class test as this was found to give better accuracy than the one versus one method. A normalised confused matrix was also plotted for this result.

4 Results

4.1 Similarity Matrix

Figure X below shows the plot of the similarity matrix:

This visualisation shows christmas as being the most distinct class, having a good separation from all of the other classes. Cosmetics are also shown to be fairly distinct from every other class. The other 3 classes seem to have poor separation between each other with perfume and car being the least separated. This implies that christmas and cosmetics should perform well in binary classifications however the multi-class classification should suffer from poorer results coming from the other 3 classes.

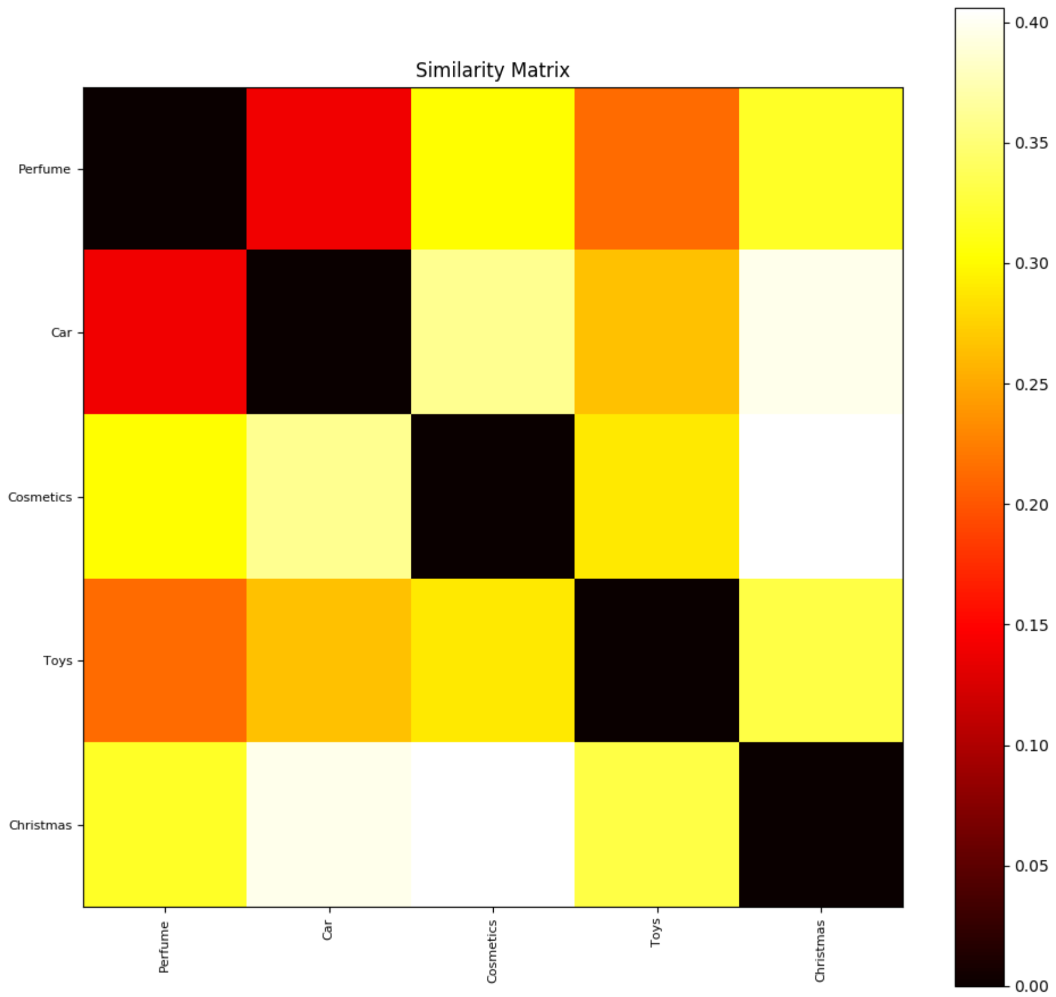

Figure 5: Figure X: Similarity matrix showing how distinct classes are from each other based on the euclidean distance between the features of each. The whiter the colour, the more seperated the classes are. The columns give the true labels and the rows give the predicted labels.

4.2 Clustering

The 2 dimensional visualisation of the 5 classes using the t-SNE method is shown in figure X. This plot shows that there are no clear distinctions between the classes. There are however some signs of clustering for some of the classes present. The most distinct of which is the christmas class which is mostly clustered in the bottom left part of the plot. This agrees with the information from the similarity matrix of this class being the most distinct. Most of the cosmetic data points are clustered at the top with the rest being evenly spread out amongst most of the other classes. There are very little cosmetic data points lying within the christmas cluster implying that these 2 classes have the most separation which is also shown in the similarity matrix. The toy class is mostly spread although it has formed somewhat of a clusters on top of the cosmetic cluster. This implies that the toy class itself will be largely mislabeled whereas while the cosmetic class is fairly distinct from the rest, it will likely have most of its mislabels be as the toy class. The other 2 classes, car and perfume, are both fairly evenly spread throughout meaning they don't have their own specific identity of features.

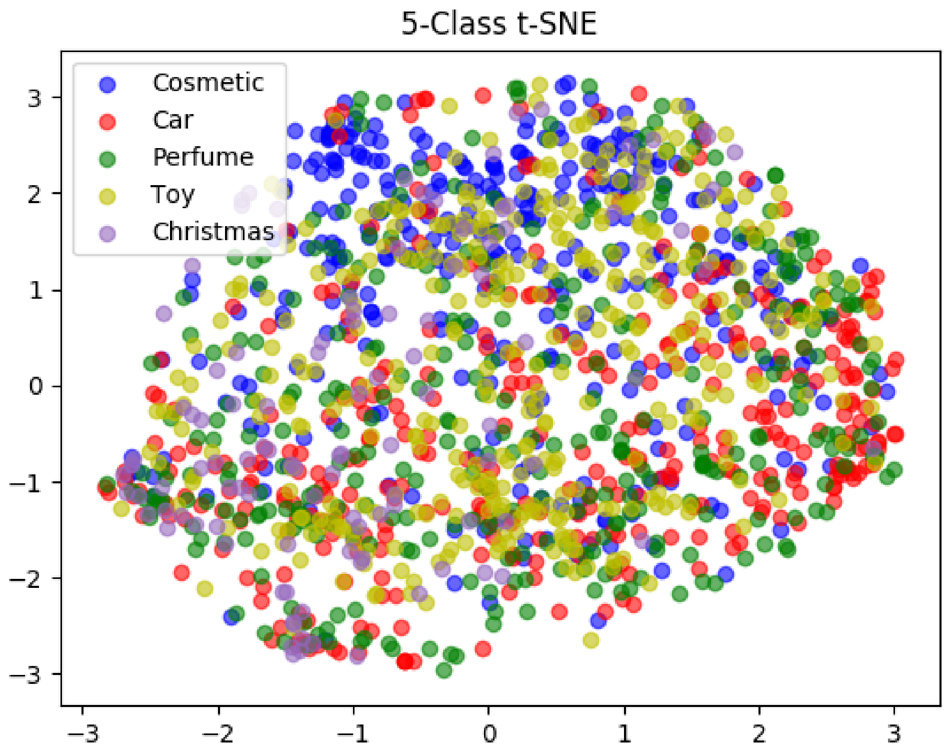

Figure 6: Visualisation of the 5 classes with the features mapped down to 2 dimensions using the t-SNE method. The axes values are arbitrary.

4.3 Classification

4.3.1 Extra Trees Classifier

The following tables are of the different scoring metrics for the 10 1 versus 1 classifications for the extra trees classifier:

Perfume	Car	Cosmetic	Toy	Christmas
Score	0.64±0.04	0.80±0.04	0.86±0.03	0.80±0.03
Precision	0.65±0.05	0.79±0.04	0.85±0.03	0.72±0.13
Recall	0.57±0.06	0.81±0.05	0.86±0.04	0.33±0.09
F1	0.61±0.05	0.80±0.04	0.85±0.03	0.45±0.09

Table 2: Table showing all of the scoring metrics for the binary classifications of perfume against the other 4 classes using the extra trees classifier.

Car	Perfume	Cosmetic	Toy	Christmas
Score	0.64±0.04	0.80±0.03	0.83±0.03	0.82±0.03
Precision	0.65±0.05	0.81±0.04	0.84±0.04	0.77±0.14
Recall	0.57±0.06	0.76±0.05	0.83±0.05	0.35±0.09
F1	0.61±0.05	0.79±0.03	0.83±0.03	0.47±0.10

Table 3: Table showing all of the scoring metrics for the binary classifications of car against the other 4 classes using the extra trees classifier.

Cosmetic	Perfume	Car	Toy	Christmas
Score	0.80±0.04	0.80±0.03	0.80±0.03	0.85±0.03
Precision	0.79±0.04	0.81±0.04	0.80±0.03	0.81±0.09
Recall	0.81±0.05	0.76±0.05	0.79±0.05	0.48±0.11
F1	0.80±0.04	0.79±0.03	0.79±0.03	0.59±0.09

Table 4: Table showing all of the scoring metrics for the binary classifications of cosmetic against the other 4 classes using the extra trees classifier.

Toy	Perfume	Car	Cosmetic	Christmas
Score	0.86±0.03	0.83±0.03	0.80±0.03	0.87±0.02
Precision	0.85±0.03	0.84±0.04	0.80±0.03	0.93±0.07
Recall	0.86±0.04	0.83±0.05	0.79±0.05	0.49±0.09
F1	0.85±0.03	0.83±0.03	0.79±0.03	0.64±0.08

Table 5: Table showing all of the scoring metrics for the binary classifications of toy against the other 4 classes using the extra trees classifier.

Christmas	Perfume	Car	Cosmetic	Toy
Score	0.80±0.03	0.82±0.03	0.85±0.03	0.87±0.02
Precision	0.72±0.13	0.77±0.14	0.81±0.09	0.93±0.07
Recall	0.33±0.09	0.35±0.09	0.48±0.11	0.49±0.09
F1	0.45±0.09	0.47±0.10	0.59±0.09	0.64±0.08

Table 6: Table showing all of the scoring metrics for the binary classifications of christmas against the other 4 classes using the extra trees classifier.

All of the binary classifications scored around 80% in all 4 metrics aside from perfume versus car and the recall values for christmas. The lower score of about 60% of the perfume versus car classification is likely due to these 2 classes having no distinct features from each other as seen in their lack of clustering in figure X. While christmas performs just as well as the other classes in score and precision, it has very low recall values against every other class. Since christmas is the class with the least number of data points, it is labeled as the positive class. This means that the classifier performs poorly when labeling christmas classes correctly but doesn't have many issues of incorrectly labeling other classes as christmas.

Next is the normalised confusion matrix from the multi-class classification:

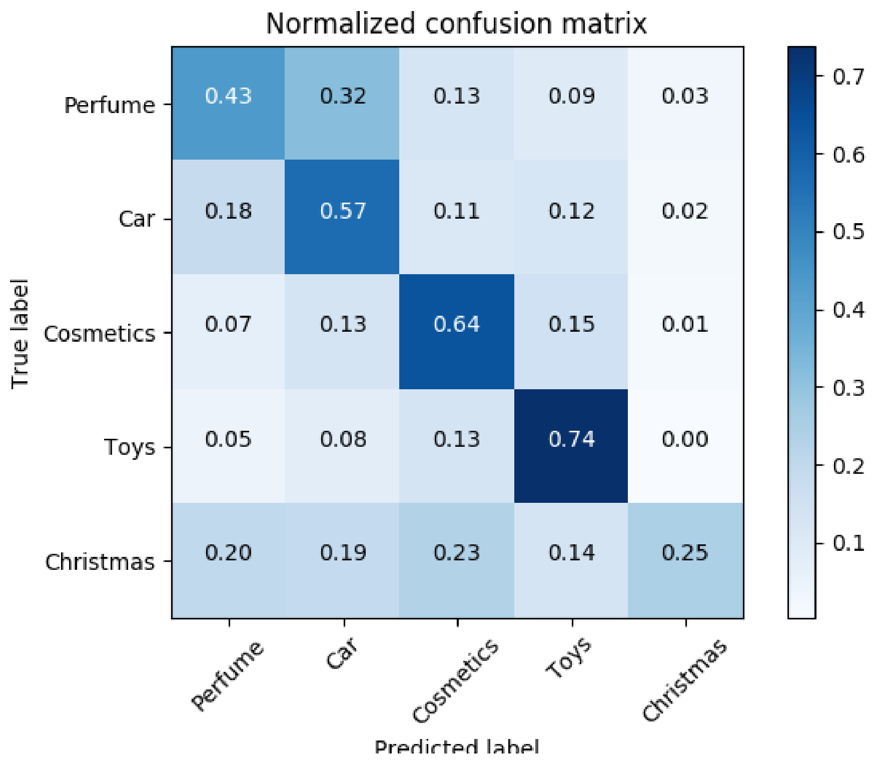

Figure 7: A normalised confusion matrix showing the percentage of labels of each class that each of the classes are given using the extra trees classifier. The main diagonal gives the correct labels and the higher the number/darker the colour of these, the most it has predicted correctly

The percentage of correct predictions for each class is reasonably lower than from the binary classifications. This is always going to be the case however as for every extra class added, there is more potential for incorrect predictions. Unlike the binary case which would have a 50% accuracy if the prediction was to be done randomly, this 5 class case would have a roughly 20% accuracy, varying slightly with the uneven number of class data points. Compared to this the classifier scored reasonably well, especially when predicting toy adverts. There are a significant number of perfume adverts being labeled as car adverts which agrees with the common theme shown throughout the other results. The most interesting result is that all of the classes had nearly 0 mislabels as christmas however christmas itself score extremely poorly, scoring no better than a random prediction. It seems that while the christmas features are good at guiding the other samples away from the christmas leaves in the decision tree, they are not enough good at guiding christmas samples towards them.

4.3.2 SVC

The following tables are of the different scoring metrics for the 10 1 versus 1 classifications for the SVC classifier:

Perfume	Car	Cosmetic	Toy	Christmas
Score	0.59±0.04	0.77±0.03	0.80±0.03	0.66±0.05
Precision	0.60±0.03	0.77±0.04	0.84±0.04	0.87±0.03
Recall	0.65±0.06	0.78±0.05	0.77±0.05	0.65±0.06
F1	0.62±0.04	0.78±0.03	0.80±0.04	0.74±0.04

Table 2: Table showing all of the scoring metrics for the binary classifications of perfume against the other 4 classes using the SVC classifier.

Car	Perfume	Cosmetic	Toy	Christmas
Score	0.59±0.04	0.79±0.03	0.81±0.03	0.71±0.04
Precision	0.60±0.03	0.78±0.04	0.81±0.03	0.90±0.03
Recall	0.65±0.06	0.82±0.04	0.81±0.05	0.70±0.05
F1	0.62±0.04	0.80±0.03	0.81±0.03	0.79±0.03

Table 3: Table showing all of the scoring metrics for the binary classifications of car against the other 4 classes using the SVC classifier.

Cosmetic	Perfume	Car	Toy	Christmas
Score	0.77±0.03	0.79±0.03	0.76±0.03	0.81±0.03
Precision	0.77±0.04	0.78±0.04	0.77±0.03	0.90±0.03
Recall	0.78±0.05	0.82±0.04	0.76±0.04	0.85±0.04
F1	0.78±0.03	0.80±0.03	0.76±0.03	0.87±0.02

Table 4: Table showing all of the scoring metrics for the binary classifications of cosmetic against the other 4 classes using the SVC classifier.

Toy	Perfume	Car	Cosmetic	Christmas
Score	0.80±0.03	0.81±0.03	0.76±0.03	0.77±0.04
Precision	0.84±0.04	0.81±0.03	0.77±0.03	0.92±0.03
Recall	0.77±0.05	0.81±0.05	0.76±0.04	0.77±0.05
F1	0.80±.04	0.81±0.03	0.76±0.03	0.84±0.03

Table 5: Table showing all of the scoring metrics for the binary classifications of toy against the other 4 classes using the SVC classifier.

Christmas	Perfume	Car	Cosmetic	Toy
Score	0.66±0.05	0.71±0.04	0.81±0.03	0.77±0.04
Precision	0.87±0.03	0.90±0.03	0.90±0.03	0.92±0.03
Recall	0.65±0.06	0.70±0.05	0.85±0.04	0.77±0.05
F1	0.74±0.04	0.79±0.03	0.87±0.02	0.84±0.03

Table 6: Table showing all of the scoring metrics for the binary classifications of christmas against the other 4 classes using the SVC classifier.

Overall the SVC scored roughly the same as the extra trees classifier of around 80% with the SVC having more variation in the results. This classifier also had the same problem of distinguishing between car and perfume adverts, implying it is not a weakness in the classifiers themselves but either a weakness in the data or that car and perfume adverts are just very similar in their audio features. This classifier did not have the problem that the extra trees did in mislabeling christmas classes, the recall values for the christmas class this time are similar to the other classes.

Figure 8 shows the normalised confusion matrix from the multi-class classification.

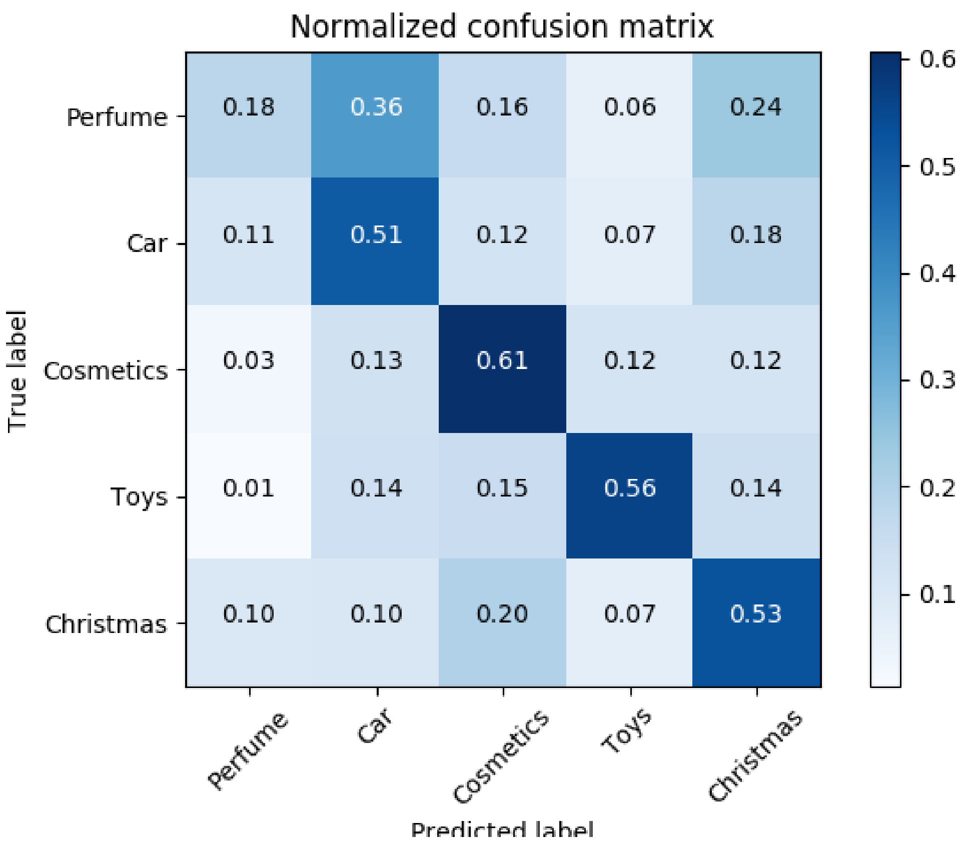

Figure 8: A normalised confusion matrix showing the percentage of labels of each class that each of the classes are given using the SVC classifier. The main diagonal gives the correct labels and the higher the number/darker the colour of these, the most it has predicted correctly

The SVC scored fairly similar to the extra trees classifier overall however the worst performing class this time is the perfume class as opposed to chstimas. The SVC has a much higher number of classes being mislabeled as christmas as well as a higher score for the christmas class implying the algorithm used for this classifier puts much more weight on the christmas data than the extra trees classifier does. The perfume adverts are mislabeled as car adverts at roughly the same amount as in extra trees however the added mislabeling of perfume adverts as christmas adverts causes it to score very poorly overall.

From all of the data it is clear that the algorithms are much better at performing binary classifications compared to multi-class. As seen in figure X, this is mostly due to the fact that while most of the pairs of classes can be seen to be distinct from each other, all of the classes together have a very high amount of overlap. While scores of around 80% can be considered to be fairly good, they are poor compared to what machine learning algorithms are capable of and are not high enough to be of much use commercially. The main contributing factors to the lower value of the scores is of the lack of data and the quality of the data. Typically machine learning is done on datasets of thousands or millions of data points and only using a few hundred is not enough for a classification algorithm to generate robust boundaries between classes. For the specific topic of adverts, there isn't much to be done about this because of the lack of adverts in general and so the main focus if looking to improve on the results would be to focus on the quality of the data. Having the data come from only a few youtube queries led to it containing a lot of noise from things such as videos that were not actually adverts or videos which contained things such as introductions before the adverts actually start. This could be improved on by looking for websites which contain lists linking to quality videos of adverts and by hand picking the samples more carefully. While this would not be doable on larger scale datasets, the low number of data points for this topic makes it more reasonable to do. Another thing that would be done to improve the scores would be to add unstructured data to the features such as the date of adverts or their country of origin.

5 Conclusion

The extra trees classifier and SVM machine learning algorithms were successful in classifying adverts from 5 different classes in binary and multi-class tests however the nature of the topic of adverts made acquisition of large amounts of quality data troublesome. To improve on the results, higher quality data must be used along with a much larger dataset however getting both of these things is a whole other challenge in itself. Visualisations were also made which showed that while some advert categories have clear distinctions between them, others could not be easily separated by their audio features alone and would need other types of features in order to make better distinctions.

References

[1] Meinard Müller, *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*, Springer, 1st edition, 2015, Pages 123-125

[2] C. Harte, M. Sandler, M. Gasser, *Detecting Harmonic Change in Musical Audio*, ACM Press, 2006, Pages 21-26

[3] N. Buduma, N. Locascio, *Fundamentals of Deep Learning: Designing Next-Generation Machine Learning Algorithms*, O'Reilly Media, 1st edition, 2017, Preface

[4] C. E. Shannon, *Communication in the Presence of Noise*, Proceedings of the IEEE, 1998, Pages 447-457

[5] V. C. Chen, H. Ling, *Time-frequency Transforms for Radar Imaging and Signal Analysis*, Artech House, 1st edition, 2002, Pages 28-31

[6] K.R. Rao, D. N. Kim, J. J. Hwang, *Fast Fourier Transform - Algorithms and Applications*, Springer, 1st edition, 2010, Page 111

[7] O. Maimon, L. Rokach, *Data Mining and Knowledge Discovery Handbook*, Springer, 2nd edition, 2010, Page 555

[8] B. Iser, G. Schmidt, W. Minker, *Bandwidth Extension of Speech Signals*, Springer, 1st edition, 2008, Page 36

[9] T. Li, M. Ogihara, G. Tzanetakis, *Music Data Mining*, CRC Press, 2nd edition, 2012, Pages 60-62

[10] T. H. Park, *Introduction to Digital Signal Processing: Computer Musically Speaking*, World Scientific, 1st edition, 2010, Pages 399-400

[11] A. Lerch, *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*, Wiley, 1st edition, 2012, Page 42

[12] L. Rokach, O. Z. Maimon, *Data Mining with Decision Trees: Theory and Applications*, World Scientific, 2nd edition, 2014, Pages 10-15

[13] P. Mather, B. Tso, *Classification Methods for Remotely Sensed Data*, CRC Press, 2nd edition, 2016, Page 188

[14] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2nd edition, 2009, Pages 226-227 and Page 417

[15] D. A. Zighed, J. Komorowski, J. Zytkow, *Principles of Data Mining and Knowledge Discovery*, Springer, 1st edition, 2003, Page 146

[16] D. Julian, *Designing Machine Learning Systems with Python*, Packt Publishing Ltd, 1st edition, 2016, Page 170

[17] L. van der Maaten, *Visualizing Data using t-SNE*, Tilburg University, 1st edition, 2008, Page 4-7