Music Genre Classification

Note:Markdown cells in notebook also explain code side-by-side.

In order to classify a song as either a rock song or a classical song, 30 features have been extracted.
Note:Bonus task is in other file.
These features include:

MFCC (12 coefficients)

MFCC takes into account human perception for sensitivity at appropriate frequencies by converting the conventional frequency to Mel Scale, and is thus useful in categorizing sound. It concisely describes the overall shape of a spectral envelope.
Even though higher order **coefficients** represent increasing levels of spectral details, depending on the sampling rate and estimation method, **12** to 13 cepstral **coefficients** are typically optimal for audio analysis. Selecting a large number of cepstral **coefficients** results in more complexity in the models. Therefore, I extract only 12 co-efficients on account that they'll provide reasonably similar accuracy with less processing required.

Spectral Centroid

The spectral centroid is a measure used in digital signal processing to characterise a spectrum. It indicates where the center of mass of the spectrum is located. Perceptually, it has a robust connection with the impression of brightness of a sound(In audio production **brightness** refers to upper mid and high frequency content. Increased levels of mid and high frequency content are referred to as **brighter).**
Example-A metal song has its spectral centroid shifted towrds the end while that for a blues song will remain in the middle.

RMS

Root mean square energy carried by the wave over some observation time. Will genreally be more for rock music.

Spectral Flatness

Spectral flatness is typically measured in decibels, and provides a way to quantify how tone-like a sound is, as opposed to being noise-like. We can expect classical music to be more tone-like.

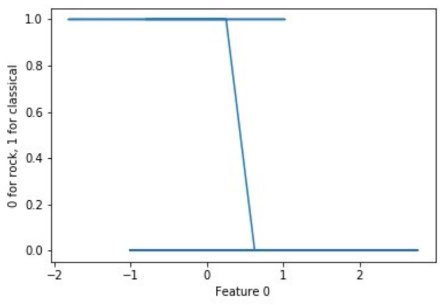
Zero Crossing Rate:

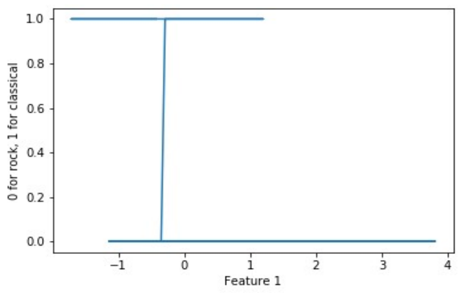
The zero-crossing rate is the rate of sign-changes along a signal, i.e., the rate at which the signal changes from positive to zero to negative or from negative to zero to positive. It has a higher value for highly percussive sounds.

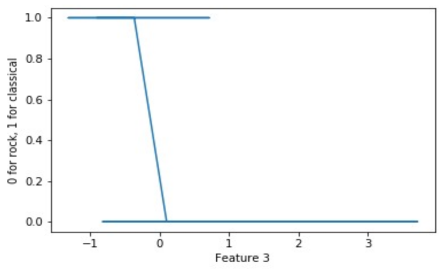
Delta (1st and 2nd order of each feature)

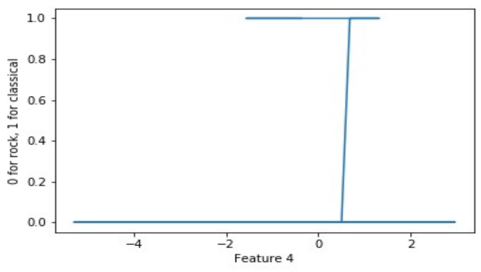
The audio was first extracted from the respective folders and placed in an empty list 'musics' genre wise.
Librosa.feature was then used to extract the above features from the audio.
Rock music was given the label '0' and Classical music was given the label '1'.

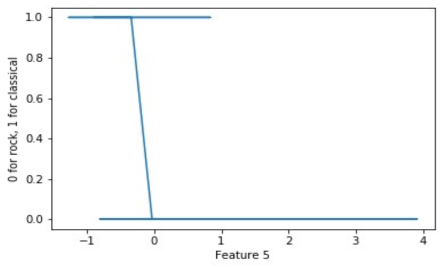
The following diagrams illustrate the variation of some such feature for the two types of music. Note that points along y=1 represent classical music and points on y=0 represent rock music.

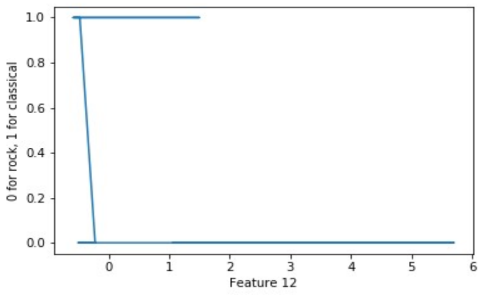






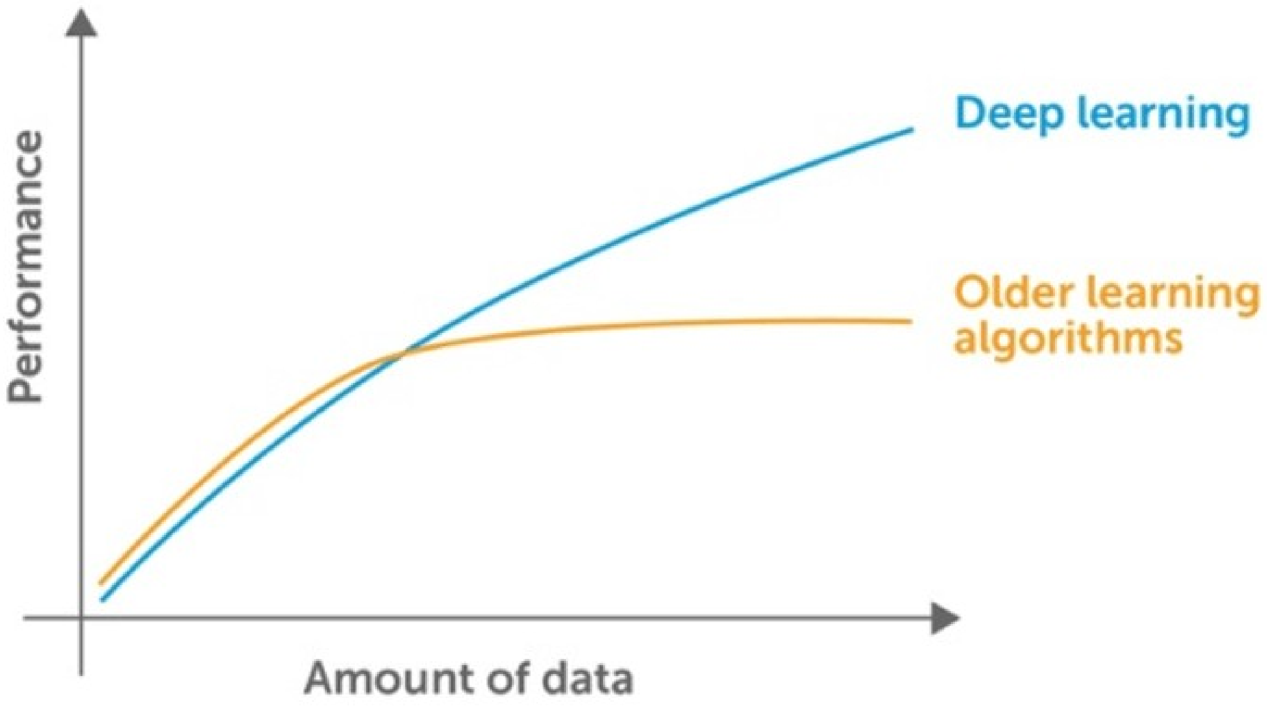






It can be seen that there is a variation in the extent and location of the values of a particular feature for rock and calssical music. It is this variation that I hope to exploit.

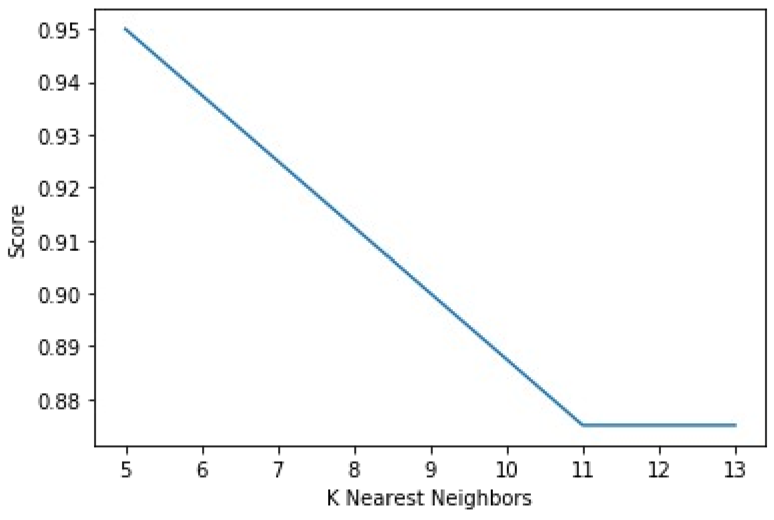
Firstly, the feature array X was scaled using standard scaler by removing the mean and scaling to unit variance.



I haven’t used a neural network based approach for this task becuase of the small dataset used. As substantiated by the above graph, a non-deep learning algortihm should perform reasonably well given the amount of data.

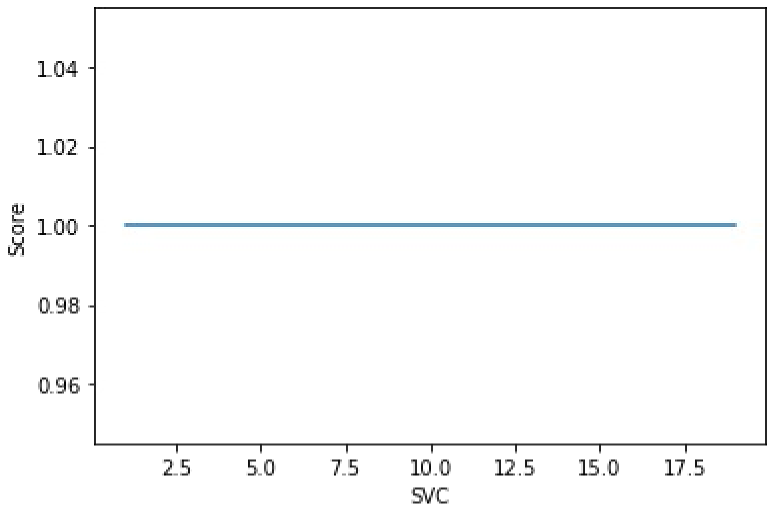
Thereafter, the first model I used on the training set (an 80-20 split) was a **KNN classifier**.

A KNN classifier works on the simple assumption that a particular data point should be allotted the same label as the mode of its k-nearest neighbours. This is simple and fairly easy to implemnt and so was my choice for a base model.
I first attempted to find the optimal k for which my accuracy would be highest.
I did this by plotting accuracy vs k for a reasonable range of k and obtained the following graph:



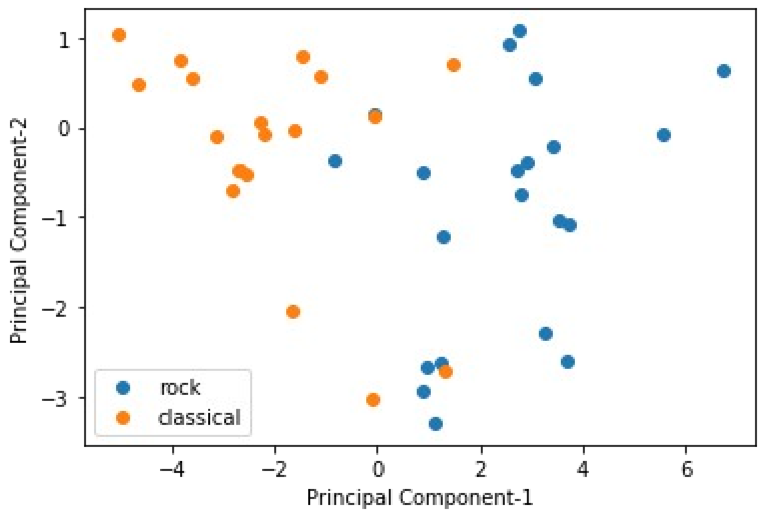
Clearly, the maximum accuracy I could obtain was 0.95, which, though good, couln’t be improved upon further by tuning hyperparameter values.

I then chose to implement an SVM Suppport Vector Classifier with a linear kernel since it generally performs better than ordinary logistic regression. Again, to estimate the best possible value of hyperparameter C, I plotted the variation of accuracy against C and obtained the following graph:



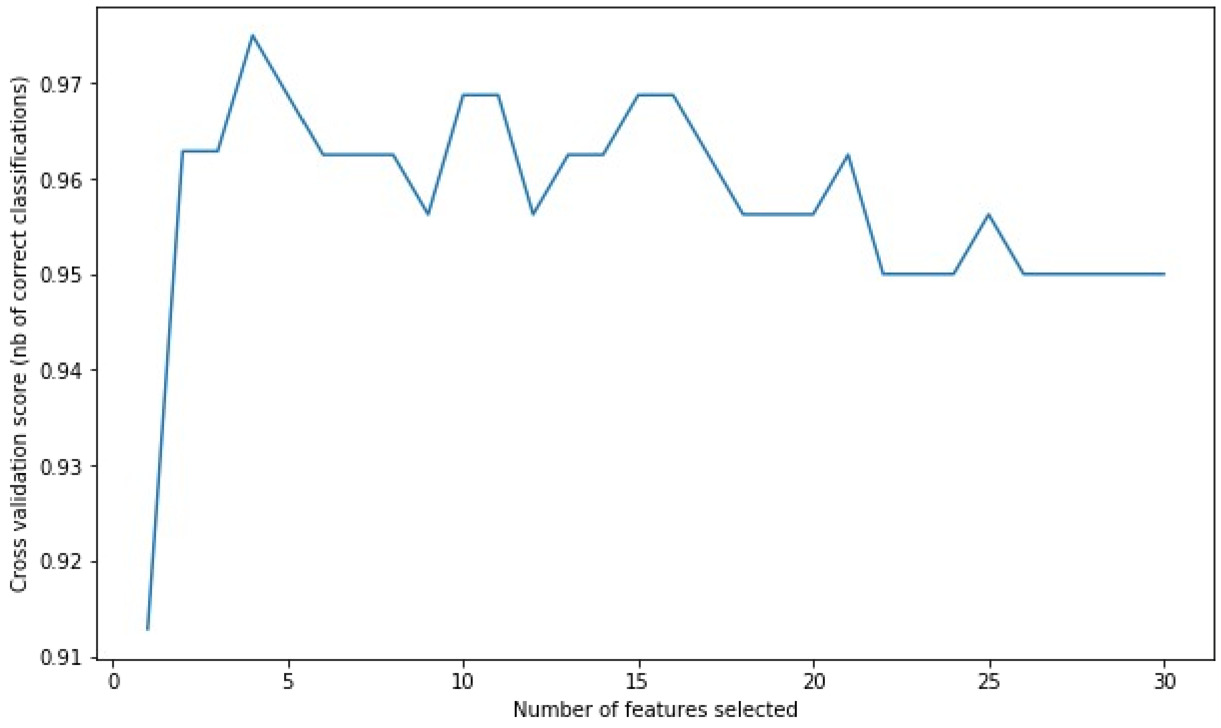
Here, I achieved an accuracy of 1.0, i.e., all songs classified correctly.

But this might not really be a perfect model. There is definitely a fear of overfitting and the 100% result might actually be misleading and fail on a larger dataset.
To improve our prediction by reducing the possibility of overfitting, I will be introducing a cross-validation set and also try to see if a lesser number of features does the job better.
Before that, I plot a PCA graph of the current situation that illustrates the two distinction between the two genres in an easy to visualize form:



RFECV allows me to choose an optimal number of features.

It checks the accuracy of a model with a given set of features on a cross validation set taken from the training set and return the set of features that provided the maximum accuracy.
Recursive feature elimination is basically a backward selection of the predictors. This technique begins by building a model on the entire set of predictors and computing an importance score for each predictor. The least important predictor(s) are then removed, the model is re-built, and importance scores are computed again.
The model I have passed to RFECV is a linear kernel SVM, since that was what gave the 1.0 accuracy on the training set.



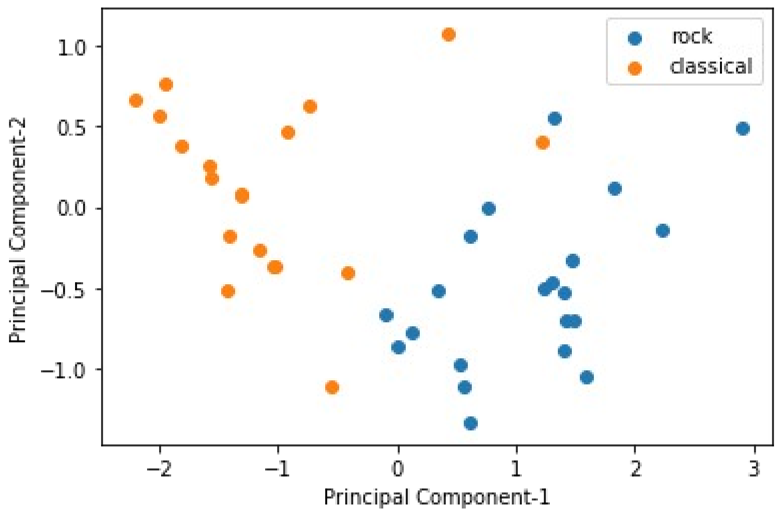
```
Optimal number of features: 4
Selected features: [3, 5, 15, 24]
```

4 features namely, (3,5,15,24) have provided maximum prediction accuracy on the cross validation set. In fact, using all 30 features actually gives a low accuracy of around 0.95 on the cross-validation set-an indication that our previous answer of 1.0 on the training set was a sign of overfitting.

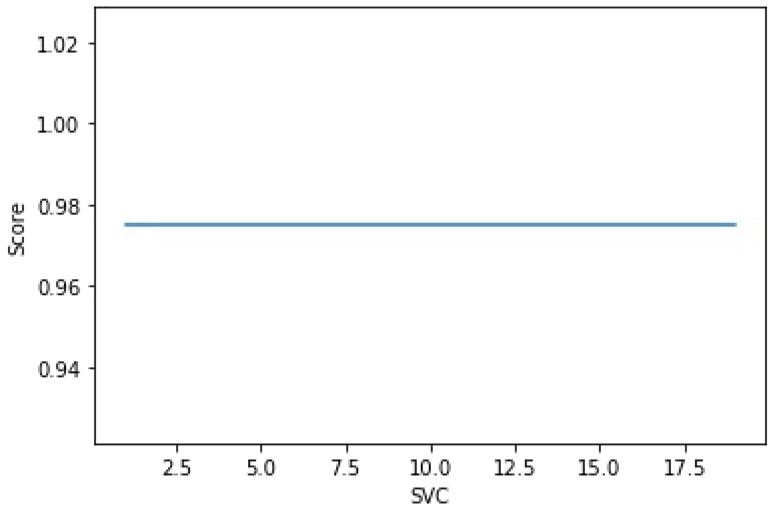
However, there is still one thing I can do.
A correlation matrix gives me the extent to which different features are correlated to each other.
Using two correlated features together in the model makes it redundant.
On printing out the features with correlation>0.8, I find that two of my selected features, namely, 3 and 5 are correlated to each other.

However, on printing ou the importance of all 4 features in my model, both 3 and 5 have a significant contribution.
Also, the dropping of one variable doesn’t do much to increase our processing efficiency since the number of features is already less. Both features are highly related to output. Therefore, I keep both of them.

Having done this, I now plot a PCA graph of the dataset using these 4 features and scaling them down to two for easy visualization.
Again, we can see the distinction between the two genres.



Now, once again, I aim to find an optimal C value for my SVC, this time to be trained on four features.



With all C's giving the same accuracy, I choose any C, say C=9, and train myfinal model on the reduced set of features.

Finally, I obtain an accuracy of 0.975 on the training set. This indicates a single misclassified example.
Though this seems a step back from the original result of 1.0, it actually isn't becuase we have avoided overfitting the dataset-something which was being done by the previous model. This model will work better on general datasets.

On running the model on the fiven given songs, the following classification is obtained:
Song-1
```
0.024960 rock
0.975040 classical
```

```
Song-2
0.999273 rock
0.000727 classical

Song-3
0.013630 rock
0.986370 classical

Song-4
0.933143 rock
0.066857 classical

Song-5
0.157724 rock
0.842276 classical
```

(*Code and result in notebook)*

VIBHHU SHARMA
EE19B128

Note:I had also tried outlier detection and removal for a ’better’ model. However, after removing the outliers (using IQR), despite giving an accuracy of 1.0 on cross-validation, only an accuracy of 0.9 was achieved on a test. This was an indication wherein in removing these ‘outliers’, you would be removing perfectly valid data points. More investigation is required before their removal.