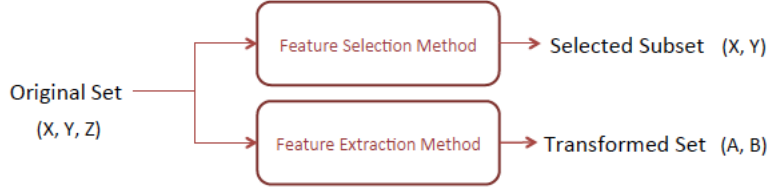
Music Mood Classification

## Feature Selection

Feature selection allows us to understand and visualize better our data. It reduces the training time and also the computational cost and use of memory. The most relevant benefit is that it improves the prediction.



We can either only select some features, or we can transform our features into a new space (for instance, PCA).

We can use different ways to generate our subset, but in this lab we are going to focus on the evaluation of it, using filter and wrapper methods.

**Filters:**

Measures relevance. Robust against overfitting but might not be the best solution to find useful features.

**Wrappers:**

Measures feature usefulness to a given predictor.

Prone to overfitting.

## Feature Importances

In order to complete this exercise, we have computed the feature importances in 3 different ways. Find the explanation below.

First of all we have loaded our data and we have preprocessed it. First we replace all nan’s and infs by 0. We have had issues because some NaN were written by hand with a space after, and pandas didn’t read it as NaN. So when cleaning the NaN, these values written by hand were not cleaned. We have extracted the different classes, and we have encoded them so that they become numerical. This is the mapping: 'aggressive': 1, 'happy': 2, 'relaxed': 3, 'sad': 4

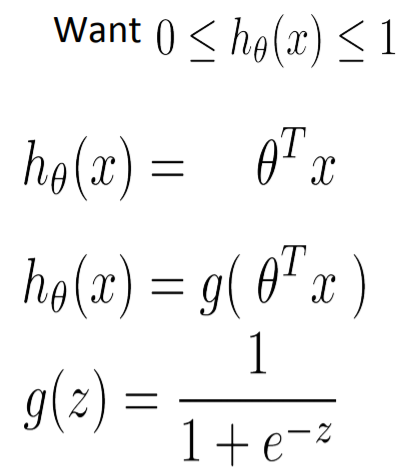
We also have splitted train and test into X and target. This has been done to facilitate the implementation of algorithms and handling with the data.

We have 66 predictors, and notice that the title of the song has become the index.

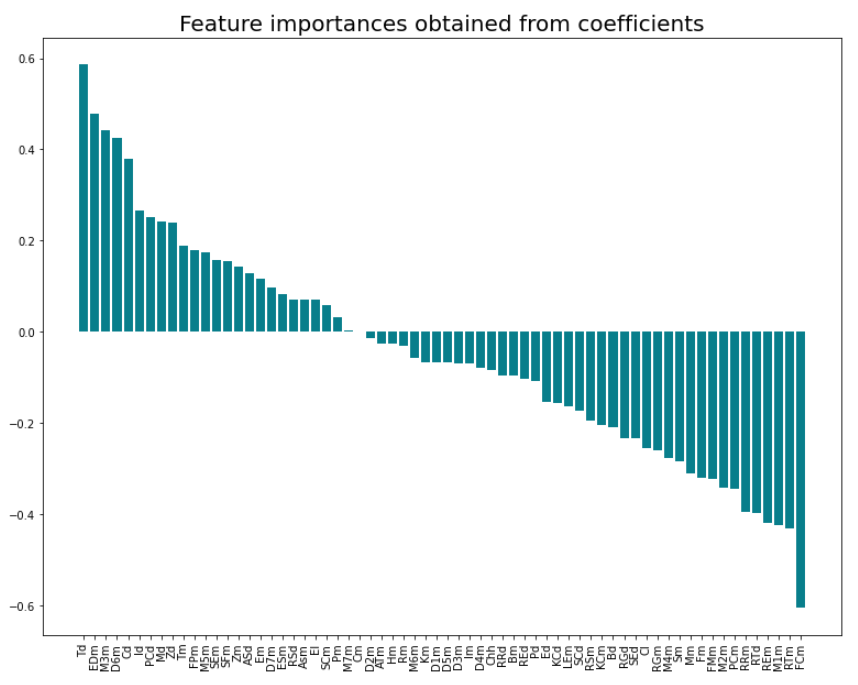
All of the values are numeric, and there are no missing values. The only obvious problem is the scale. Just take a look at the mean area and mean smoothness columns — the differences are drastic, which could result in poor models. That is the reason why we scale all the data.

Using coefficients

We are going to examine the model’s coefficients. We are going to use logistic regression. As we know from the theory, this is an equation containing the coefficients of each input variable, in other words, the importance.



Once we apply the model, we get the following graph:



Large coefficients (they can be positive or negative) mean that the feature is relevant, hence it has some importance when we apply a classification model. If the coefficient is close to 0, it means that the variable doesn’t have an impact on the prediction.

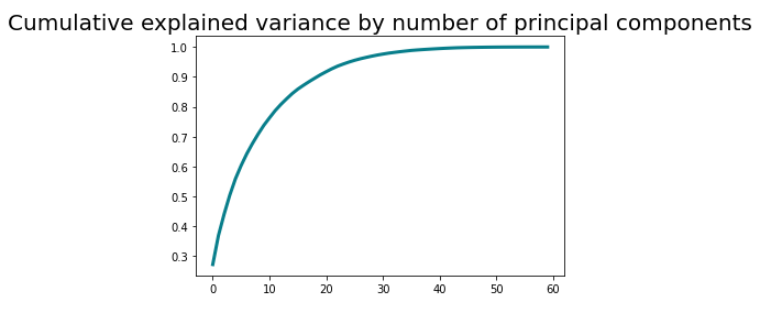
In summary, the larger the coefficient, the more influential it is on a prediction.

This means that Td, EDm, FCm and RTm, for instance, are the most relevant features, that is, rhythm and structure.

Using PCA

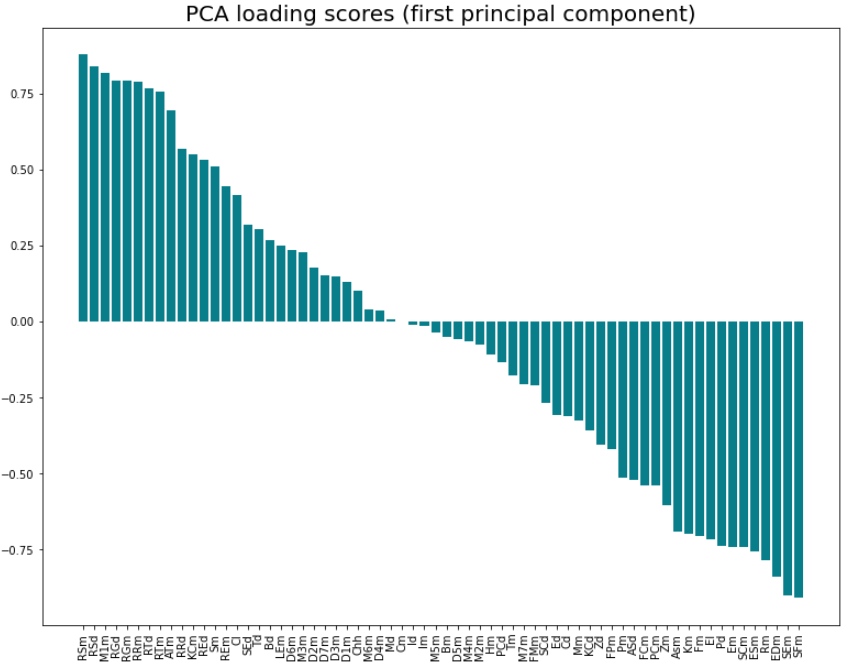
This is a feature extraction method. It allows us to reduce the dimensions of the data based on feature importance. In this case, we are not going to get the most important features directly like in the previous case. We are going to get the N principal components.

What we do in the code is apply PCA to our scaled data.



In this figure, we can see that 90% of the variance is explained using approximately the first 18 principal components.

Since this is a little bit hard to understand, we compute the correlations between our real features and the principal components.



This plot shows the relationship between all the input variables and the first principal component.

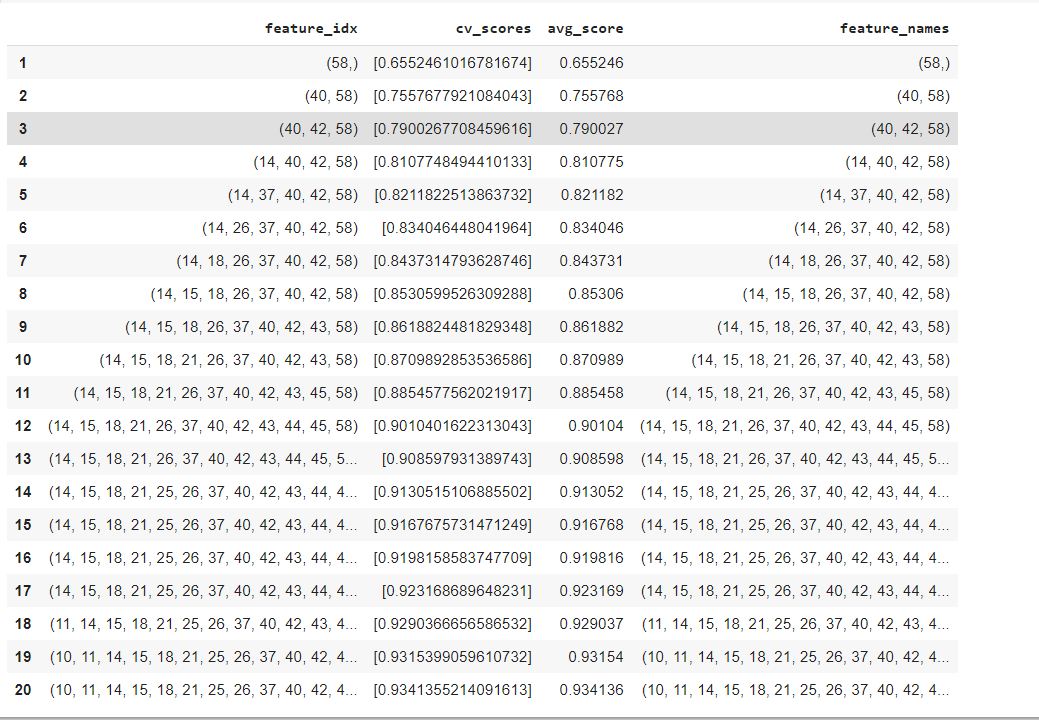
This is basically telling us in which direction the data “moves”. The features with value 0, mean that they are not contributing in this component. Notice that the one contributing the most would be RSm. This implies that, the larger the values of RSm are for a given register, the larger values would take this first component. The same way would happen with SFm. If SFM takes very large negative values, the 1st component would also do it.

To sum up, when we do a PCA we are grouping variables and putting them on top of a plane to give them a sense (that are the eigenvectors). Hence, we can have positive and negative contributions. This implies that when we want to group features, we would take ones or others depending on whether we are interested in explaining the positive or negative part. In this case we can see that most features with a large positive number in the first principal component belong to structure while some of the large negative belong to timbre. Among the largest values, we can find features belonging to the timbre.

Using Sequential Feature Selection

This is a greedy search algorithm and it is part of the wrapper methods. It finds the optimal feature subset by iteratively selecting features based on the classifier performance. We start with an empty feature subset and add one feature at the time in each round; this one feature is selected from the group of features that are not in our feature subset, and it would help result in the best classifier performance.

We have used a maximum subset length of 20 features, taking into consideration that there are 66, we think it’s a good number.



These are the subsets obtained. We can see that the features selected for the largest subset belong to all the categories except dynamic.

Using Variance Threshold

This is a filter method. We compute the variance of each feature, and we select the subset of features based on a user-specified threshold. In our case, the threshold has been set to 0.4, so we are going to keep all the features with variance >= to 0.4 We work under the assumption that higher variance means more useful information. But we are not taking into account the relationship between these variables and the target. This is one of the drawbacks of this method.

Once we apply this algorithm we are left with a subset of 26 features that are non-constant. These are:

['Em', 'Ed', 'El', 'LEm', 'ATm', 'Asm', 'ASd', 'EDm', 'FPm', 'FMm', 'FCm', 'Tm', 'Td', 'PCm', 'PCd', 'Pm', 'Pd', 'Cm', 'Cd', 'Cl', 'Chh', 'KCm', 'KCd', 'Mm', 'Md', 'Hm', 'ESm', 'Rm', 'Im', 'Id', 'Bm', 'Bd', 'SCm', 'SCd', 'Zm', 'Zd', 'Sm', 'Km', 'SEm', 'SEd', 'SFm', 'Fm', 'REm', 'REd', 'M1m', 'D1m', 'M2m', 'D2m', 'M3m', 'D3m', 'M4m', 'D4m', 'M5m', 'D5m', 'M6m', 'D6m', 'M7m', 'D7m', 'RSm', 'RSd', 'RRm', 'RRd', 'RTm', 'RTd', 'RGm', 'RGd']

Here, we can find features belonging to all the categories.

## Improvements

We should validate the subsets obtained, or in case of the PCA, the new subset transformed in order to check if they improve our prediction. We could apply the same classifier used to generate our actual data but with the new subsets and see whether the metrics such as the accuracy, recall, etc. improve with respect to using all the 66 features.

In conclusion, in all the cases we have obtained that the relevant features belong to all the categories. This matches our initial belief: music is very complex and there are many factors involved that determine its mood.

## Error Analysis

We have decided to train an ANN with our data. In order to validate our subset, we have tested the network with all the features and then also we have tested it with only the subset obtained from sfs.

First of all, we have scaled our data using min\_max\_scaler and then we have passed our y column to dummy (otherwise the network was not working).

We have built the network using Keras instead of Torch because it’s more “user-friendly”.

We have obtained a 55-60% accuracy using all the features, whereas using the subset obtained by SFS we obtain an accuracy of 50%.

Hence, our subset could be improved.

Also, to improve our model we could have used cross-validation to test different sizes of the network and different hyper-parameters (although this has been done by hand by us).

These are the songs were our classifier has failed:



It confuses some happy songs with aggressive songs, this could be due to the fact that the rhythm of both is accelerated, for instance, or the dynamics are similar. It confuses some relaxed with sad, we think that for the same reason: similar rhythm, etc.

It doesn’t make that much sense that it confuses happy with relaxed or sad (in terms of the rhythm, so maybe this is because of the timbre or harmony, something that to us, that haven’t studied music does not seem a difference).

We think that the accuracy of a human would be slightly different (which does not imply worse), more subjective. We have feelings and emotions and music activates zones of our brains that make us feel. Probably a human would have classified the song taking into account how he feels when listening to it (which maybe has to do with the structure of the song, the harmony, the pitch, etc. From some studies we know that high-frequency sounds make us get on the alert). The thing is that an algorithm does not have feelings so it bases its classification purely on real values of these metrics.

## Bibliography / code guides used:

<https://stackabuse.com/applying-filter-methods-in-python-for-feature-selection>

<https://sebastianraschka.com/faq/docs/feature_sele_categories.html>

<https://towardsdatascience.com/feature-selection-for-machine-learning-in-python-wrapper-methods-2b5e27d2db31>

<https://towardsdatascience.com/3-essential-ways-to-calculate-feature-importance-in-python-2f9149592155>