Chord progressions selection based on song audio features

Noelia Rico[1] and Irene Díaz[1]

Computer Science Department at University of Oviedo, Spain
noripa@gmail.com, sirene@uniovi.es

Abstract. A chord progression is an essential building block in music. In the field of music theory is usually assumed that these progressions influence the mood, emotion, genre or other critical aspects of the songs, and also in the perception that they will cause on the listener. Therefore, it is natural to think that musical and audio features of a track should be related to its chord progressions. Choosing carefully these progressions when it comes the time of creating a new song, is a fundamental aspect depending on the feelings we want to evoke on the listener. Also, two songs can be considered alike or classified into the same emotions or genres if they use the same chord progressions. Many music classification studies are presented nowadays, but none of them take into account chord progressions, probably due to the lack of this kind of data. In this paper, classification algorithms are used to illustrate the influence of the songs' features when it comes to pick up chord progressions to create a new song.

Keywords: Music Information Retrieval, Chord Progressions, Data Mining, Machine Learning, Classification

1 Introduction

It is generally accepted that music makes us feel a wide range of emotions. The mechanisms through which music evokes feelings and impacts in our brain is a vast field of research with a lot of unanswered questions.

On many occasions, the songs can be stirring even though they do not have any lyrics, and this is why music is considered an international language. In these cases, the songs' emotions are purely related to their structure [1]. Structural features of a song include rhythm, musical nuances, melody and the chords over which these melodies are settled. The same song can convey entirely different emotions after changing its chord progressions. Thus, listeners' feelings after hearing the modified song could be completely opposite. Those changes might also modify other aspects of the song such as its energy or popularity.

In the field of music regarding chord progressions, previous work has only focused on the analysis of music scores (most of the time using a MIDI file) in order to automatize the process of generating new music. Some of the previous research in this discipline is based on the identification of the chords and the

progressions themselves [2]. Others, center their efforts in using machine learning techniques to generate music chords progression, based on songs that are used to build the model and taking into account only their musical score [3–5]. Related work in this field of study combines musical features with the song lyrics [6] to classify the songs or use specific musical features to divide the songs into emotions [7].

However, there has been little discussion on how the audio features and the impact of the song on the listener are correlated with the chord progressions found in the song. One question that needs to be asked is how the chosen progressions for the composition of each song influence some musical aspects not directly related to the music score, such as the energy that the songs transmit or their future popularity.

This paper proposes a methodology to find the audio features of a track affecting their chord progressions. These audio features can be described as the musical features of the song in addition to some other attributes about its impact on the listener.

This paper is divided into five sections. Section 2 describes the dataset, explaining its creation and describing its features in detail. Section 3 gives an overview of the methods used to build the models. In section 4 the results of the machine learning process are analyzed. Finally, the last section discusses the evaluation and possible application of the classification process and future work is proposed.

2 Datasets creation

One of the most complex tasks in this work is getting the data set. For this purpose, data from Hook Theory [8] and Spotify [9] websites are considered. The information about a song provided by these two different sources is supposed to be related one to each other. Thus, the two datasets obtained are joined in order to perform a better prediction. Then, this section first details how to obtain information from Hook Theory and Spotify websites and how to put them together.

2.1 Hook Theory

Hook Theory [8] is a popular website in the music field that allows their users to analyze songs regarding chord progressions and share their analysis. It is designed to be used by people who are learning how to compose new music, integrating an interactive interface for people who do not know anything about music theory. For this purpose, this website includes a section showing the most popular chord progressions in music of all styles, ages and genres. Those "most popular progressions" are grouped into categories, to make easier the learning process for the amateur composers.

Using the API [10] of the website we have gathered all the songs that contain a specific song progression. The chord progressions categories that have been

used for this work are shown in Table 1. The second column describes the name used on the website to identify each progression and the first one indicates the group where the progression is wrapped, regarding how challenging is to use it when composing a song. The third column represents the chord progression itself and the fourth column shows the name that has been given to each progression in this project, and therefore the one which is used to identify the variable representing the progression in the final dataset.

The third column represents the chord progression as the succession of musical chords expressed by Roman numerals, as is common in Classical music theory. Following the standard, the capital letters are used to express whether the chord is major or minor. The progressions with lower difficulty are built with chords corresponding to basic degrees in the scale (I, V, iii...). Whilst difficulty increases, the chords evolve to more complex constructions, and therefore the appearance of indexes becomes popular to indicate the inversion of the chord.

Difficulty	**Name**	**Chord Progression**	**Dataset variable**
Beginner	Most popular progression	$I - V - vi - IV$	most.popular
	Pachelbel's progression	$I - V - vi - iii$	pachelbel
	Effective in all genres	$vi - V - IV - V$	effective.all.genres
	Those magic changes	$I - vi - IV - V$	magic.changes
	Gaining popularity	$I - IV - vi - V$	gaining.popularity
	Timeless	$I - V - IV - V$	timeless
Intermediate I	The cadential $^{6}_{4}$	$I^{6}_{4} - V$	cadential64
	Stepwise bass down	$I - V^{6} - vi$	stepwise.bass.down
	Stepwise bass up	$I - ii^{7} - I^{6}$	stepwise.bass.up
	The newcomer	$I - iii^{6}_{4} - vi$	newcomer
	I^{6} as a precadence	$IV - I^{6} - V$	I.as.precadence
	Simple yet powerful	$IV - I^{6} - ii$	simple.yet.powerful
Intermediate II	Chord pleased the Lord	$V - V^{6}/vi - vi$	chord.pleased.Lord
	Expanding with V/vi	$I - V/vi$	expanding.V.vi
	vi^{4}_{2} as a passing chord	$vi - vi^{4}_{2} - IV$	vi42.passing.chord
	Secondary dominant	$V^{7}/IV - IV$	secondary.dominant
	Applied vii°	$vii^{\circ}/vi - vi$	applied.vii
Advanced	Cadencing in style	$IV - iv - I$	candencing.in.style
	A mixolydian cadence	$\flat VII - IV - I$	mixolydian.cadence
	Using ♭ VI to set up a V	$\flat VI - V$	bVI.to.V
	Cadencing via ♭ VII	$\flat VII - I$	cadencing.bVII

Table 1: Attributes obtained from the Hook Theory API

Once all the songs have been collected, the dataset is build using a row to represent each song. Songs are identified by two columns, which are title and artist, and the dataset contains 21 more columns, each one associated to

a different chord progression. If the song of the row i contains the progression j, the $cell_{ij}$ has the value 1, otherwise its value will be 0. The final dataset extracted from Hook Theory website [8] contains 1990 songs characterized by 23 attributes.

2.2 Spotify

The Spotify API fetches data [9] of all the available songs in the Spotify music catalog. To construct the second dataset based on Spotify, the songs contained in HookTheory dataset are considered. For each song in the hooktheory dataset, a search with the Spotify API is made using the primary key of the songs (artist and title). If the track is found, their features are extracted and the song is added to this new dataset. Among all the data available in the Spotify API, only the attributes that have been considered useful for this research will be included. Some of these features are strictly pulled out of the song's musical attributes like tempo or key, but others are calculated from its audio features like valence, or based on their reproductions on Spotify like popularity. These features are used as predictors of the final dataset. A thorough description of the attributes can be found in Table 2.

2.3 Features and class of the final dataset

Using the data obtained from both Spotify and Hook Theory websites, different features characterizing 1990 songs have been collected. Features extracted from Spotify (see 2) represent the attributes associated with each song, while features obtained from Hook Theory website represent the class to learn. Thus, Features extracted from Spotify will be used to predict each chord progression as a binary problem (whether the song contains a chord progression or not).

Note that 21 different datasets will be created, each one linked with a different chord prediction 1. To sum up, 21 binary classification problems will be solved, taking into account the Spotify features as predictor variables. All the problems are independent one to each other, but they have the same structure. For each problem, a different binary class variable will be considered. The aim of each problem is to classify the songs into two possible categories, 0 or 1. The value will be 1 if the song uses the chord considered as the class variable for that concrete problem and 0 otherwise.

3 Learning Procedure

3.1 Methods

The classification will be made using `R`, and more precisely the `caret` [11] package. Six different algorithms will be used to build the models. This will allow us to make a comparison about the performance and the final results:

- `rpart`: PART tree

feature	type	description
artist	chr	used as id of the song together with title
title	chr	used as id of the song together with artist
popularity	int	The value will be between 0 and 100 expressing the popularity of the track, with 100 being the most popular. Songs that are being played a lot now will have a higher popularity than songs that were played a lot in the past
explicit	Factor	Two possible levels: TRUE/FALSE. Whether or not the track has explicit lyrics
danceability	num	Value between 0.0 and 1.0 describing how suitable a track is for dancing based on a combination of musical elements (tempo, rhythm stability, beat strength, overall regularity). A value of 0.0 is least danceable and 1.0 is most danceable
energy	num	Value between 0.0 to 1.0 that represents a perceptual measure of intensity and activity. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy
key	Factor	12 possible levels. The key track is in. Integers map to pitches using standard Pitch Class notation
loudness	num	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track. Values typical range between -60 and 0 db
speechiness	num	Value between 0.0 to 1.0. Detects the presence of spoken words in a track. The more exclusively speech-like the recording the closer to 1.0 the attribute value
acousticness	num	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic
instrumentalness	num	Value between 0.0 and 1.0. The closer the instrumental ness value is to 1.0, the greater likelihood that the track contains no vocal content. Vocals like "Ooh" and "aah" are treated as instrumental in this context
valence	num	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive while tracks with low valence sound more negative
tempo	num	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
duration_ms	int	The duration of the track in milliseconds
time_signature	int	An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure)
mode	Factor	Mode indicate the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0

Table 2: Features fetched from the Spotify API

– `glm`: Generalized Linear Model
– `nb`: Naive Bayes
– `svmLinear`: Support Vector Machines with Linear Kernel
– `svmRadial`: Support Vector Machines with Radial Basis Function Kernel
– `ranger`: Random Forest

The performance and utility of the above-detailed Machine Learning methods in different domains ([12], [13], [14], [15], [16]) allows us to use them for predicting chord progressions. Different methods have been selected to cover different learning paradigms.

Method `glm` [**?**] fits generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution. It is suitable for binary classification problems.

Tree modeling is performed through `rpart` [17] and `ranger` [18] (upgrade of the classical `randomForest`). `rpart` carries out a CART (Classification and regression trees) modeling. First of all, it grows the (classification in this case) tree until one of the possible stop conditions are reached: the number of observations in a node is under a threshold, the minimum cost complexity factor cannot be reached when attempting to do a split or the tree has reached the maximum depth allowed. Once the tree has been grown, the method examines the results and if over fitting is detected, the tree is pruned to find an optimal point between over fitting and under fitting. These methods split the data based on how the creation of sub-nodes increases the homogeneity of those resultant sub-nodes. Random forests technique is based on trees too but it improves predictive accuracy by generating not one but a large number of bootstrapped trees using random samples of variables for each iteration, classifying a case using each tree in this newest created forest, and deciding the final predicted outcome by combining the results across all of the trees.

Naive Bayes is based on the Bayes Theorem and although all the predictors are assumed to be independent within each class label, it gives excellent results for many real problems. The predictor variables are handled by assuming that they follow a Gaussian distribution, given the class label. The outcome variable is predicted using the probabilities of each attribute when `nb` [19] is applied. Support Vector Machines [20] performs classification looking for the hyperplane that maximizes the margin between the classes closest points. When a linear separator cannot be found, data points are projected into a higher dimensional space where the data become linearly separable. This projection is made by kernel techniques. Two different methods with linear `svmLinear` and non-linear `svmRadial` kernel have been chosen

The above introduced methods have been applied considering the following settings:

– Method configuration: Default parameter settings provided by caret package in order to perform a first approach of how the datasets behave with each algorithm.
– Validation: Stratified split into training (80%) and test (20%). The training set will be used to build the model and the test set to evaluate it.

– Training: 10 fold cross validation with 3 repetitions has been used for training the models. In this 10 fold cross validation, all the samples in the dataset were grouped in 10 different sets, using one of them for testing while keeping the remaining for training. This process is repeated three times.

Note that most of the datasets are imbalanced (30%(1)/70%(0)). Thus, different sampling strategies are tested and the resulting datasets represent the input to each one of the learning methods. For each of the six algorithms, four different models have been built according to a different sampling strategy (`over`, `down`, `ROSE` and `SMOTE` [11]). For each algorithm, the model yielding the best results on the test set will be selected as the best model. The evaluation of this previously unseen instances of the test set let us compare the real performance of the final models. Algorithm 1 shows the whole procedure.

```
algorithms = {glm, rpart, randomForest, nb, svmLinear, svmRadial};
foreach dataset do
    foreach alg in algorithms do
        ; /* train the models                                    */
        modOver = train(alg, data = train, sampling = over);
        modDown = train(alg, data = train, sampling = down);
        modROSE = train(alg, data = train, sampling = ROSE);
        modSMOTE = train(alg, data = train, sampling = SMOTE);
        ; /* predict the class: test set                        */
        predOver = predict(modOver, data = test);
        predDown = predict(modDown, data = test);
        predROSE = predict(modROSE, data = test);
        predSMOTE = predict(modSMOTE, data = test);
        ; /* evaluate the models                                 */
        resOver = logLoss(confMatrix(test$class, predOver));
        resDown = logLoss(confMatrix(test$class, predDown));
        resROSE = logLoss(confMatrix(test$class, predROSE));
        resSMOTE = logLoss(confMatrix(test$class, predSMOTE));
        best = maxMicroF1(resOver, resDown, resROSE, resSMOTE);
    end
end
```

Algorithm 1: Build and select the best model for each dataset

3.2 Metrics

As explained before, some of the datasets' class are imbalanced. In addition to Accuracy (A), the micro averaged version of the well known metrics Precision (P), Recall (R) and F_1 [11] is selected together with logLoss [21]. Considering that TP is the number of examples correctly classified as the positive class, TN is the number of examples correctly classified as the negative class, FP is the

number of examples wrongly classified as the positive class and FN the number of examples wrongly classified as the negative class, A, P, R and F_1 are defined as follows.

$$A = \frac{TP + TN}{TP + TN + FP + FN} \qquad P = \frac{TP}{TP + FP}$$
$$R = \frac{TP}{TP + FN} \qquad F_1 = 2 \times \frac{P \times R}{P + R}$$

LogLoss quantifies the accuracy of a classifier by penalizing wrong classifications. Minimizing the Log Loss is equivalent to maximizing the accuracy of the classifier, but there is a subtle difference. In order to calculate the Log Loss, the classifier must assign a probability (p_i) to each class rather than simply yielding the most likely class. It is defined below.

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}[y_i log p_i + (1 - y_i) log(1 - p_i)]$$

4 Results

progression	rpart	glm	nb	svmL	svmR	rf
effective.all.genres	0.74825	0.69578	0.73521	0.69849	0.74934	**0.79193**
gaining.popularity	0.84074	0.72467	**0.83956**	0.75179	0.82217	0.80793
magic.changes	**0.86839**	0.76379	0.29982	0.72214	0.82843	0.86310
most.popular	0.67680	0.65479	0.65490	0.67097	**0.69075**	0.68522
pachelbel	0.74137	0.86923	0.83064	0.87494	**0.90270**	0.75444
timeless	0.73474	0.75117	0.41014	0.68525	**0.82318**	0.56898
cadential64	0.83682	0.81701	0.62892	0.82006	0.88455	**0.89908**
I.as.precadence	**0.95234**	0.79190	0.83517	0.75889	0.81397	0.76771
newcomer	0.77865	0.81385	0.68669	0.82314	0.74944	**0.91111**
simple.yet.powerful	**0.94063**	0.76104	0.51293	0.80395	0.91206	0.92766
stepwise.bass.down	0.71599	0.69285	0.76457	0.70821	0.72704	**0.78109**
stepwise.bass.up	0.48695	0.84957	0.63792	0.89845	**0.92008**	0.82191
applied.vii	0.82918	0.80775	0.81376	0.83822	**0.88389**	0.74348
chord.pleased.Lord	**0.88399**	0.85250	0.41050	0.82230	0.78972	0.63735
expanding.V.vi	0.81187	0.70590	**0.84821**	0.72716	0.81105	0.80814
secondary.dominant	0.90437	0.88067	0.85573	0.88628	**0.94234**	0.91330
vi42.passing.chord	0.89668	0.82953	0.86953	0.83557	**0.91233**	0.82343
bVI.to.V	0.84705	0.85576	0.81376	0.83221	**0.90814**	0.66306
cadencing.bVII	0.80866	0.73626	**0.83309**	0.75970	0.82873	0.82798
candencing.in.style	0.84614	0.78441	**0.93802**	0.82860	0.88073	0.93221
mixolydian.cadence	0.90957	0.79328	0.49135	0.76125	0.84047	**0.90957**

Table 3: F_1 for each model on test set

progression	rpart	glm	nb	svmL	svmR	rF
effective.all.genres	2.29080	0.61709	1.06604	0.64638	0.54090	**0.50819**
gaining.popularity	2.29586	0.59268	0.68104	0.55032	0.56741	**0.53052**
magic.changes	0.45711	0.88835	5.69869	0.58640	0.45529	**0.41536**
most.popular	0.63065	0.61723	0.92931	0.61045	0.61680	**0.59226**
pachelbel	0.73712	0.57187	2.20434	0.49506	**0.42239**	0.66887
timeless	1.95744	0.85849	4.27938	0.64156	**0.45389**	0.74188
cadential64	0.55121	0.66113	3.76041	0.51237	**0.36370**	0.39110
I.as.precadence	0.76120	0.63053	1.47563	**0.56608**	0.61571	0.62741
newcomer	0.67570	1.00073	2.20718	0.49264	0.80781	**0.39696**
simple.yet.powerful	1.38791	0.73773	2.11598	0.60028	**0.32238**	0.39084
stepwise.bass.down	2.03425	0.64715	0.96973	0.64085	0.61982	**0.60725**
stepwise.bass.up	1.14947	0.50541	1.76185	0.51010	**0.34447**	0.60753
applied.vii	0.69399	0.61890	1.22321	0.58839	**0.40091**	0.66001
chord.pleased.Lord	**0.39493**	2.57653	10.74308	0.46906	0.85578	1.33383
expanding.V.vi	0.55673	0.66775	**0.47183**	0.65301	0.49104	0.59195
secondary.dominant	0.36368	0.39952	0.77317	0.35223	**0.17554**	0.26221
vi42.passing.chord	0.66081	0.62432	0.88796	0.50365	**0.34007**	0.58048
bVI.to.V	1.78251	0.51041	3.98525	0.47484	**0.32722**	0.72091
cadencing.bVII	0.78476	0.60006	0.72928	0.56762	**0.49911**	0.45051
candencing.in.style	0.58210	0.61831	0.53075	0.44616	0.38625	**0.38411**
mixolydian.cadence	1.59762	0.63714	3.21393	0.53388	0.52423	**0.40109**

Table 4: LogLoss of each model on the test set

In this section, the aforementioned experiments are analyzed. Table 3 shows the F_1 obtained for each binary problem and each different machine learning method. Table 4 contains the LogLoss value for the experiments whose F_1 is shown in Table 3.

Note that F_1 is often high. However, the highest F_1 is obtained with the combination of `SMOTE` resampling method and `svmRadial`. In particular, the results with highest F_1 and lowest logLoss are obtained when model is trained with `svmRadial`, being this the best for the 38% of the progressions, followed by the method `ranger`, which achieves the best results for the 33% of the progressions. We recommend the use of `svmRadial` because it has lower computational cost, and therefore it is quicker building the model. Table 5 shows the most important variables used for building the models. It should be noticed how the *valence* appears in the top four important variables for most of the models. Other features such as *danceability*, *valence*, *loudness* or *popularity* together with other strictly musical related like *instrumentalness* or *key*, stand out from the list of important variables used to build the models.

The evidence from this study points towards the idea that information gathered from Spotify could be useful for helping us decide which chord progressions should be used when a new song is being created.

progression	first	second	third	fourth
effective.all.genres	duration_ms	acousticness	energy	instrumentalness
gaining.popularity	duration_ms	loudness	valence	speechiness
magic.changes	tempo	acousticness	danceability	valence
most.popular	loudness	duration_ms	valence	popularity
pachelbel	key8	valence	mode1	instrumentalness
timeless	mode1	valence	acousticness	key1
cadential64	acousticness	speechiness	valence	duration_ms
I.as.precadence	valence	acousticness	danceability	duration_ms
newcomer	acousticness	mode1	valence	danceability
simple.yet.powerful	valence	danceability	popularity	instrumentalness
stepwise.bass.down	valence	popularity	acousticness	duration_ms
stepwise.bass.up	acousticness	energy	loudness	key10
applied.vii	acousticness	instrumentalness	energy	valence
chord.pleased.Lord	loudness	speechiness	popularity	energy
expanding.V.vi	speechiness	valence	loudness	instrumentalness
secondary.dominant	energy	loudness	acousticness	valence
vi42.passing.chord	key6	acousticness	loudness	valence
bVI.to.V	loudness	valence	energy	danceability
cadencing.bVII	popularity	loudness	instrumentalness	acousticness
candencing.in.style	acousticness	loudness	popularity	instrumentalness
mixolydian.cadence	loudness	instrumentalness	acousticness	valence

Table 5: Most important variables of each model

5 Conclusions and Future Work

This paper presents a method to improve knowledge about how useful chord progressions can be in other music classification problems, which aims to classify songs into emotions or genres.

The method presented in this work has many interesting applications in the composition of new music. If done manually, the work of analyzing all the chord progressions in a music score is not an easy task. It requires time and a great knowledge of music theory. Automating this process is not trivial either, it has a high computing cost, and the results are not always satisfactory.

These results represent an excellent initial step toward a new field of application. The principal advantages are the creation of new songs based on the features of similar and already existing songs, without the need to analyze their structure and taking into account not only their musical features but aspects of the tracks as their energy or valence.

We are currently working on the multi-label version of this problem, that can address the issue of creating a new track based on a bunch of songs we want it to be similar to. Given as input existing songs, the classification model will return the list of chord progressions that should and should not be used to create the new composition.

Further work will look into the lyrics of the songs, in order to perform a sentiment analysis process to explore the correlation between the Spotify musical

features, the chord progressions and the emotions the song intends to cause on the listener, based on the vocabulary they are using.

Acknowledgments

This research has been funded by the Spanish MINECO project TIN2017-87600-P.

References

1. Juslin, P.N., Sloboda, J.: Handbook of Music and Emotion: Theory, Research, Applications. Oxford University Press (2011)
2. Cho, Y.H., Lim, H., Kim, D.W., Lee, I.K.: Music emotion recognition using chord progressions. In: Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on, IEEE (2016) 002588–002593
3. Stamatatos, E., Widmer, G.: Automatic identification of music performers with learning ensembles. Artificial Intelligence **165**(1) (2005) 37 – 56
4. Arutyunov, V., Averkin, A.: Genetic algorithms for music variation on genom platform. Procedia Computer Science **120** (2017) 317 – 324 9th International Conference on Theory and Application of Soft Computing, Computing with Words and Perception, ICSCCW 2017, 22-23 August 2017, Budapest, Hungary.
5. Costa, Y.M., Oliveira, L.S., Silla, C.N.: An evaluation of convolutional neural networks for music classification using spectrograms. Applied Soft Computing **52** (2017) 28 – 38
6. Hu, X., Downie, J.S.: Improving mood classification in music digital libraries by combining lyrics and audio. In: Proceedings of the 10th Annual Joint Conference on Digital Libraries. JCDL '10, New York, NY, USA, ACM (2010) 159–168
7. Martin-Gómez, L., Navarro-Cáceres, M.: Applying data mining for sentiment analysis in music. In De la Prieta, F., Vale, Z., Antunes, L., Pinto, T., Campbell, A.T., Julián, V., Neves, A.J., Moreno, M.N., eds.: Trends in Cyber-Physical Multi-Agent Systems. The PAAMS Collection - 15th International Conference, PAAMS 2017, Cham, Springer International Publishing (2018) 198–205
8. Famous-Chord-Progressions: https://www.hooktheory.com/theorytab/common-chord-progressions (January 2018)
9. Spotify: https://developer.spotify.com/web-api/get-audio-features/ (January 2018)
10. HookTheory-API: https://www.hooktheory.com/api/trends/docs (January 2018)
11. Kuhn, M.: Building predictive models in R using the caret package. Journal of Statistical Software **28**(5) (11 2008) 1–26
12. Villar, J.R., Chira, C., Sedano, J., González, S., Trejo, J.M.: A hybrid intelligent recognition system for the early detection of strokes. Integrated Computer-Aided Engineering **22**(3) (2015) 215–227
13. Herrero, Á., Sedano, J., Baruque, B., Quintián, H., Corchado, E., eds.: 10th International Conference on Soft Computing Models in Industrial and Environmental Applications, SOCO 2015, Burgos, Spain, June 2015. Volume 368 of Advances in Intelligent Systems and Computing., Springer (2015)
14. Troiano, L., Rodríguez-Muñiz, L.J., Ranilla, J., Díaz, I.: Interpretability of fuzzy association rules as means of discovering threats to privacy. Int. J. Comput. Math. **89**(3) (2012) 325–333

15. Gil-Pita, R., Ayllón, D., Ranilla, J., Llerena-Aguilar, C., Díaz, I.: A computationally efficient sound environment classifier for hearing aids. IEEE Trans. Biomed. Engineering **62**(10) (2015) 2358–2368
16. Montañés, E., Quevedo, J.R., Díaz, I., Ranilla, J.: Collaborative tag recommendation system based on logistic regression. In: Proceedings of ECML PKDD (The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases) Discovery Challenge 2009, Bled, Slovenia, September 7, 2009. (2009)
17. Chambers, J.M.: Statistical Models in S. CRC Press, Inc., Boca Raton, FL, USA (1991)
18. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth and Brooks, Monterey, CA (1984)
19. Wright, M.N., Ziegler, A.: ranger: A fast implementation of random forests for high dimensional data in C++ and R. Journal of Statistical Software **77**(1) (2017) 1–17
20. Rish, I.: An empirical study of the naive bayes classifier. In: IJCAI 2001 workshop on empirical methods in artificial intelligence. Volume 3., IBM New York (2001) 41–46
21. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3) (Sep 1995) 273–297
22. Masnadi-shirazi, H., Vasconcelos, N.: On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In Koller, D., Schuurmans, D., Bengio, Y., Bottou, L., eds.: Advances in Neural Information Processing Systems 21. Curran Associates, Inc. (2009) 1049–1056