

## ¿If we had to do it again¿ - an algorithmic view of the magic formula behind a commercially successful French hip-hop song

**Auteur :** Zolotariov, Denis

**Promoteur(s) :** Ittoo, Ashwin

**Faculté :** HEC-Ecole de gestion de l'Université de Liège

**Diplôme :** Master en ingénieur de gestion, à finalité spécialisée en digital business

**Année académique :** 2022-2023

**URI/URL :** <http://hdl.handle.net/2268.2/18786>

---

### *Avertissement à l'attention des usagers :*

*Tous les documents placés en accès ouvert sur le site le site MatheO sont protégés par le droit d'auteur. Conformément aux principes énoncés par la "Budapest Open Access Initiative"(BOAI, 2002), l'utilisateur du site peut lire, télécharger, copier, transmettre, imprimer, chercher ou faire un lien vers le texte intégral de ces documents, les disséquer pour les indexer, s'en servir de données pour un logiciel, ou s'en servir à toute autre fin légale (ou prévue par la réglementation relative au droit d'auteur). Toute utilisation du document à des fins commerciales est strictement interdite.*

*Par ailleurs, l'utilisateur s'engage à respecter les droits moraux de l'auteur, principalement le droit à l'intégrité de l'oeuvre et le droit de paternité et ce dans toute utilisation que l'utilisateur entreprend. Ainsi, à titre d'exemple, lorsqu'il reproduira un document par extrait ou dans son intégralité, l'utilisateur citera de manière complète les sources telles que mentionnées ci-dessus. Toute utilisation non explicitement autorisée ci-avant (telle que par exemple, la modification du document ou son résumé) nécessite l'autorisation préalable et expresse des auteurs ou de leurs ayants droit.*

---



## “IF WE HAD TO DO IT AGAIN”

AN ALGORITHMIC VIEW OF THE MAGIC FORMULA BEHIND A COMMERCIALY SUCCESSFUL  
FRENCH HIP-HOP SONG

Jury:  
Supervisor:  
M. Aswhin ITTOO  
Reader :  
M. Chuor PORCHOURNG

Master Thesis submitted by  
**Denis ZOLOTARIOV**  
For the degree of  
**Master in Digital Business**

Academic year 2022/2023

## Chapter 1

# Acknowledgements

First of all, I would like to sincerely thank HEC Liège for the level of education I received during my university studies, and the incredible opportunity to learn so many diverse things, whether in class, through projects or through this master thesis.

The first persons I would like to thank are my teachers **Messrs. Ittoo & Schyns**, who supported my idea in the early stages of reflection and that have given many useful pieces of advice for the successful completion of this thesis. Thanks to my supervisor **Mr. Ittoo**, in particular, for his trust and for the autonomy he gave me, which pushed me to overcome the plethora of issues I encountered, to surpass myself and to be proud of my accomplishments.

Thanks also to **Mr. Poumay**, who kindly answered my LDA-related questions when I felt really stuck.

On a more general note, I would like to take this opportunity to thank the teachers who were able to captivate us, the students, and make us learn, as well as all the people who contribute to making HEC and the University of Liège places conducive to study and fulfillment.

In my personal life, too, I would like to thank all the acquaintances with whom I exchanged on my thesis, who gave me support, ideas or pieces of advice. Here, I obviously think of **Elsa, Aliocha, Anouar, Dimitri, Delil, Laura, Roman, Anthony, Alexia, Justine, Martin, Laureen, Delphine, Benjamin, Cédric, Victoire, Véronique, Pauline, Annelyse, Bérénice, Julien, Sorenza, Arthur & Julien**. I cannot forget also people that were so enthusiastic about my thesis (surprisingly for me) that asked me to read it when it is over: **Anissa, Adin, Dylan, Laoro, Nicolas, Mateo**. I thank you very much for your encouragement and interest.

To conclude, the achievement of this thesis took much longer than I expected, but I met a lot of people, I discovered many things and I learned during the whole process, nonstop. I even learned more about myself, my ability to deal with cumulative workload, with stress and to respect my limits. I surely would have loved to achieve even more, but I realise that I gave my best at each step without ever quitting and that I have attained a very decent result.

For all of that, I would like to express my gratitude.

## Chapter 2

# Table of contents

# Contents

<b>1</b>	<b>Acknowledgements</b>	<b>1</b>
<b>2</b>	<b>Table of contents</b>	<b>3</b>
<b>3</b>	<b>Figures and source codes</b>	<b>6</b>
<b>4</b>	<b>Introduction</b>	<b>11</b>
4.1	Justification of the topic . . . . .	12
4.1.1	The place of Internet in our lives . . . . .	12
4.1.2	The place of music in our lives . . . . .	13
4.1.3	The place of Hip-Hop in the global music industry . . . . .	14
4.1.4	The place of France in Hip-Hop . . . . .	15
4.2	Research objective and detailed research question . . . . .	15
4.2.1	A correlation model . . . . .	16
4.2.2	A prediction model . . . . .	16
4.3	Envisaged method . . . . .	16
4.3.1	Success definition and limitations . . . . .	16
4.3.2	Database creation and data to store . . . . .	16
4.3.3	Hypotheses testing and model improvement . . . . .	16
4.3.4	Prediction model . . . . .	17
4.4	Outline . . . . .	17
<b>5</b>	<b>Developments</b>	<b>18</b>
5.1	Theoretical framework . . . . .	19
5.1.1	Success: definition and limitations . . . . .	19
5.2	Methodology . . . . .	20
5.2.1	Type of research . . . . .	20
5.2.2	Data collection and analysis . . . . .	21
5.2.3	Scientific literature review . . . . .	41
5.2.4	Problems encountered and solutions found . . . . .	45
5.3	Results . . . . .	62
5.3.1	Making predictions . . . . .	64
5.3.2	Storing our model . . . . .	64
5.4	Discussion . . . . .	65
<b>6</b>	<b>Conclusions</b>	<b>66</b>
6.1	Limitations . . . . .	67
6.2	Future research paths . . . . .	67
<b>7</b>	<b>Appendices</b>	<b>69</b>
7.1	Developments . . . . .	70
7.1.1	Methodology . . . . .	70
7.2	Conclusions . . . . .	84
7.2.1	Future research paths . . . . .	84

7.3 Additional appendices . . . . .	86
<b>8 List of resource persons</b>	<b>91</b>
<b>9 Bibliography and references</b>	<b>93</b>

## Chapter 3

# Figures and source codes



# List of Figures

5.1	Example of a Genius page within this album . . . . .	22
5.2	Our <i>artists.txt</i> file . . . . .	24
5.3	Our <i>artists.txt</i> file, filtered . . . . .	25
5.4	Preview of the Diamond certified releases after LDA model running . . . . .	32
5.5	Model comparison without weighting average . . . . .	39
5.6	Model comparison with oversampling . . . . .	40
5.7	Feature importance analysis of our first model version . . . . .	44
5.8	Feature importance of model version 2 - taking release month into account . . . . .	44
5.9	Feature importance of model version 3 - taking release day into account . . . . .	45
5.10	Our first heatmap result, which is completely incorrect . . . . .	46
5.11	Our second heatmap, with correlation matrix . . . . .	47
5.12	Heatmap, third version: less features, removal of certification levels . . . . .	48
5.13	Updated topic labels in MongoDB . . . . .	52
5.14	A result sample based on our incorrect code . . . . .	53
5.15	Result sample with variable selection for model training fixed . . . . .	53
5.16	Result from a small example code written on an online Python compiler (Programiz) . . . . .	55
5.17	Misuse of <i>tokens</i> list . . . . .	56
5.18	All topics we had after topic labeling . . . . .	57
5.19	Preview of our topics replacement list . . . . .	59
5.20	Our gigantic database with features dummified . . . . .	61
5.21	Sample of our <i>.csv</i> prediction file . . . . .	64
7.1	Sample of our statistical model analysis . . . . .	80
7.2	Our initial k-fold cross-validation result with Decision Tree (overfit) . . . . .	82
7.3	Our k-fold cross-validation result with Random Forest (overfit) . . . . .	83
7.4	Our final result of k-fold cross-validation with Random Forest and the oversampling technique applied . . . . .	83
7.5	Sample of our scientific literature database on Notion . . . . .	87
7.6	Sample of our scientific literature database on Google Spreadsheet . . . . .	88
7.7	Sample of our main page for the research thesis on Notion . . . . .	88
7.8	Overview of our MongoDB database . . . . .	89
7.9	An overview of our Platinum MongoDB collection . . . . .	89
7.10	Overview of our GitHub page for the research thesis (available at [GitHub, 2023]) . . . . .	89
7.11	By-hand correction to some releases . . . . .	90

# List of Tables

5.1	Classification report: oversampling RF + day/month/year variable split . .	38
5.2	Random songs comparison for both topic modelling algorithms . . . . .	58
5.3	Results of the statistical tests . . . . .	63
5.4	Results of Chi-square realized on <i>topic_x</i> and <i>certification_level</i> variables .	63
7.1	Classification report: hard Voting Classifier (LogReg, RF & SVC) . . . . .	71
7.2	Classification report: hard Voting Classifier (DT, RF & SVC) . . . . .	71
7.3	Classification report: hard Voting Classifier (RF, SVC & KNN) . . . . .	72
7.4	Classification report: hard Voting Classifier (RF, SVC & NB) . . . . .	72
7.5	Classification report: hard Voting Classifier (RF, DT, SVC, NB & KNN) .	73
7.6	Classification report: soft Voting Classifier (RF,DT, SVC, NB & KNN) . .	73
7.7	Classification report: soft Voting Classifier (RF, SVC & NB) . . . . .	74
7.8	Classification report: soft Voting Classifier (LR, RF & SVC) . . . . .	74
7.9	Classification report: soft Voting Classifier (DT, RF & SVC) . . . . .	75
7.10	Classification report: soft Voting Classifier (RF, SVC & KNN) . . . . .	75
7.11	Classification report: oversampling RF . . . . .	76
7.12	Classification report: oversampling RF + month/year variable split . . . . .	76
7.13	Classification report: oversampling RF + day/month/year variable split . .	77
7.14	2001, 2011 and 2021 best-selling rap albums comparison . . . . .	85

# List of source codes

5.1	Retrieving all artist names through Genius . . . . .	22
5.2	Constructing the artist name's list . . . . .	23
5.3	Cleaning furthermore the artist name's list . . . . .	24
5.4	Retrieving certified albums through SNEP . . . . .	26
5.5	Retrieving album's content . . . . .	28
5.6	Pre-processing data before storing . . . . .	29
5.7	Latent-Dirichlet Allocation (LDA) topic modelling . . . . .	31
5.8	Running our LDA algorithm and storing results in variables . . . . .	32
5.9	Storing topic modelling result in JSON file . . . . .	33
5.10	Storing album data in MongoDB . . . . .	34
5.11	Creating Pandas' DataFrame . . . . .	35
5.12	Pre-processing of the DataFrame's data . . . . .	36
5.13	Defining our heatmap . . . . .	37
5.14	Results of our heatmap . . . . .	37
5.15	Double-split method to train our model . . . . .	38
5.16	Our Python code for our incorrect first heatmap . . . . .	46
5.17	Our second heatmap . . . . .	47
5.18	Correct code for the heatmap . . . . .	48
5.19	Repetitive patterns for words in initial LDA version . . . . .	49
5.20	Our program for word2vec topic modelling . . . . .	50
5.21	Modifying MongoDB's data to correct topic modelling . . . . .	51
5.22	Replacing values in <i>topic_1</i> , <i>topic_2</i> and <i>topic_3</i> . . . . .	51
5.23	Our initial training model variable selection . . . . .	52
5.24	Our variable selection, fixed . . . . .	53
5.25	Fixing the duplicated topic columns . . . . .	55
5.26	Correct bit of code for pre-processing text contents . . . . .	56
5.27	Full code for themes grouping and filtering . . . . .	60
5.28	Making featurings a binary variable . . . . .	62
5.29	Predicting the level of certification a record will get . . . . .	64
5.30	Saving our model in a <i>.joblib</i> file . . . . .	65
7.1	Declaration of the Random Forest algorithm with oversampling technique . . . . .	70
7.2	Declaration of the Decision Tree algorithm with oversampling technique . . . . .	70
7.3	Declaration of the Gaussian Naive Bayes algorithm with oversampling technique . . . . .	70
7.4	Declaration of the K-nearest Neighbors algorithm with oversampling technique . . . . .	78
7.5	Declaration of the Logistic Regression algorithm with oversampling technique . . . . .	78
7.6	Declaration of the Support Vector Classifier algorithm with oversampling technique . . . . .	78
7.7	SMOTE oversampling . . . . .	79
7.8	<i>imblearn</i> 's undersampling . . . . .	79
7.9	SMOTEENN sampling . . . . .	79
7.10	One example of Voting Classifier model . . . . .	80
7.11	Feature importance analysis code . . . . .	80
7.12	Our <i>array_dummiifier_topics()</i> function . . . . .	81

7.13	Addition to <i>array_dummi fier_topics()</i> to deal with conflicting column names	81
7.14	Our function <i>join_labels()</i> , working in pair with <i>array_dummi fier_topics()</i>	82
7.15	Output of the Decision Tree k-fold cross-validation method . . . . .	82
7.16	Initial output of the Random Forest k-fold cross-validation method . . . . .	83
7.17	Our final output for the oversampled Random Forest k-fold cross-validation method . . . . .	84
7.18	Our k-fold cross-validation algorithm taken from [Affiah, 2022] . . . . .	87

## Chapter 4

# Introduction

## 4.1 Justification of the topic

### 4.1.1 The place of Internet in our lives

We live in a digital world. We are connected more than ever before, we spend more time than ever on the Internet for always more reasons, whether it is for our personal leisure time or for work, for educational purposes or entertainment. A lot of what we do is digitalized, sometimes even transferring our habits from the real world to the digital one. This all results in an ever growing user base all around the world, as [Degenhard, 2023] shows: just to take the last two years, we went from 4,852 millions of users in 2021 to 5,100 millions the next year, a 5% increase for a technology soon to turn 30. The same source even goes as far as predicting this trend will only keep up, reaching 5,300 millions in 2023 and no less than 6,192 millions in 2028.

The impact and importance of the Internet in our lives is therefore undeniable.

What do all these users do on the Internet? According to a research conducted by [Petrosyan, 2023b] on a population of 16 to 64 years old<sup>1</sup>, when we re-group results by topics and order them by the highest ranked one first, the main activities seem to be:

1. **Learning** : searching and finding information (1), keeping up-to-date with news and events (3), researching how to do things (5), etc.
2. **Social interactions** : staying in touch with friends and family (2), etc.
3. **Creativity purposes** : finding new ideas or inspiration (6), etc.
4. **Entertainment** : watching videos, TV shows or movies (4), accessing and listening to music (7), filling up spare time and general browsing (8), etc.
5. **Commercial purposes** : researching products and brands (8), etc.

”Accessing and listening to music” represents the answer of 44% of the participants.

[Ceci, 2023b], in turn, focuses on the worldwide users activity but on smartphones in a one-year span, going from July 2022 to June 2023, on a population whose age goes from 18 to 64 years old<sup>2</sup>. Among all respondents, the top 5 activities and their respective percentage are listed here below:

1. 74.7% of respondents use their smartphone to chat or send messages
2. 70.95% of respondents use them for e-mailing
3. 62.9% for online banking
4. 61.7% to listen to music
5. 61.1% to watch videos

And these results are interesting. Indeed, we observe that communication and learning are predominant reasons we use technology, which is not surprising, but also that **music seems to have a very important place in our day-to-day lives**. It even reaches fourth place in the study conducted by [Ceci, 2023b], ahead of video watching, which is a massive market nowadays.

---

<sup>1</sup>Unfortunately, nor the population size or segmentation is available to our knowledge.

<sup>2</sup>Here too, unfortunately, we don't have much indication on the population size or segmentation.

### 4.1.2 The place of music in our lives

One way to understand how considerable the music market is, is by using the observations we have just discussed and given by [Degenhard, 2023]; if we take 5,100 million as the number of Internet users we currently have and multiply it by the percentage of respondents of any of the last two researches we discussed regarding their music consumption, we observe that we had **at least 2,244 million music listeners** (and 3,162 million listeners if we take the second results) in the world in 2022, which is a massive market.

Now, considering smartphone activities is of interest, as [Petrosyan, 2023a] demonstrated that we preferably access the Internet through any type of phones in 95% of cases, and only 63% of the time through any type of desktop. By carrying on in this subcategory, one could further ask what this traffic represents in terms of user number of revenues, for instance.

[Ceci, 2023a] gives us the first answer: among all downloaded Android applications on the Google Play Store during June 2023, users mostly downloaded:

1. **Social media applications:** Instagram (68.92 millions), Facebook (38.52 millions), WhatsApp Messenger (29.57 millions), etc.
2. **Video makers:** mAst (26.02 millions), Viamaker (23.77 millions), etc.
3. **Shopping applications:** Meesho (16.21 millions), etc.
4. **Music streaming application:** Spotify (16.11 millions), etc.

Spotify, the leading music streaming platform [Curry, 2023] [Götting, 2023], is the only one of its category appearing in this top 20. Not only that, but it ranks higher than very important markets, such as TV and series streaming or Telegram. This, once again, tends to prove the impact music has.

The second part of our question (that is, the importance of the sector revenue-wise) is answered by [StatistaResearchDepartment, 2023]; gaming is the best selling segment in mobile applications, revenue wise, with 225,339 millions of dollars in 2022. In the same year, music is ranked fifth most profitable segment, with \$12,028 millions in revenue. Forecasts also inform that this upward trend will most probably continue, with 17,880 millions of dollars of revenue expected in the music sector, an increase of 31% compared to 2023. [Curry, 2023], on its side, estimates music streaming revenues to \$43.4 billion in 2022.

Music can also be consumed in different ways. [Curry, 2023] highlights five of them and ranks them by the profitability in the United States in 2022:

1. Streaming (83%)
2. Physical (11%)
3. Other digital (4%)
4. Synchronized (2%)

[Smirke, 2023], in turn, gives its ranking but based on global music consumption in 2022:

1. Streaming (67%)
2. Physical (17.5%)
3. Performance rights (9.4%)

4. Other digital (3.6%)
5. Synchronized (2.4%)

This shows us that streaming has a very, very dominant position in the market, as it is by far the most profitable way of creating, selling and consuming music.

Naturally, each country has its culture, tastes and consumption habits. Thanks to [StatistaMarketInsights, 2023], we learn that in 2017, the top 5 countries in the world with the most music streaming revenues were:

1. United States: 3.924 million \$
2. United Kingdom: 916 million \$
3. China: 912 million \$
4. Germany: 792 million \$
5. France: 500 million \$

The booming of the sector is confirmed by the data of 2023:

1. United States: 10.200 million \$
2. China: 3.178 million \$
3. United Kingdom: 1.973 million \$
4. Germany: 1.442 million \$
5. France: 1.069 million \$

And [Smirke, 2023] proves it: in order, the United Kingdom, Germany and France are the most profitable European countries in the music streaming industry, with France having a increase of revenues of 7.7% compared to 2021, an increase even greater than the European average, which is 7.5%.

#### **4.1.3 The place of Hip-Hop in the global music industry**

Now that we know the financial size of the international music market, the next question would be: what is the most popular genre of music in music streaming platforms?

According to [Milkman, 2021], the result goes as such for each of the leading music streaming platforms in 2021:

- Apple: Hip-Hop (48%)
- YouTube: Hip-Hop (38%)
- Spotify: pop (44%), then Hip-Hop (24%)
- Pandora: country (35%), then pop, then Hip-hop
- Amazon Music: pop (48%), then country, then Hip-hop

This brings some interest in analyzing Hip-Hop more deeply.



#### 4.1.4 The place of France in Hip-Hop

The United States is, without a doubt, the biggest and most profitable Hip-Hop country market of all [Le Monde, 2020] [Lynne, 2022]. With the latter put aside, what would be the next country to be the most important for this culture?

Although opinions may diverge, a certain literature has suggested France as being the second biggest Hip-Hop market of all [Le Monde, 2020] [Lynne, 2022].

Arguments toward this hypothesis can be found by analysing national rankings; indeed, if we take the top 10 albums of 2022 in the 3 European countries we have spoken earlier, and if we look at the number of rap artists present in this top 10, we can observe the following:

- United Kingdom: 1
- Germany: 1
- France: 5 to 7<sup>3</sup>

For comparison's sake, during the same year, only 4 top-10 albums in United States were produced by rappers.

Such observations indicate the very unique place of Hip-Hop in France, a phenomenon that can be linked to the Toubon Law, which dates back to 1994 and that formulates the obligation to follow the quota of at least 40% of song radios to be French-spoken on private and public radios in traffic peak hours [Mourgere, 2015].

Considering it as an opportunity, a private radio company named Skyrock decides to take advantage of this situation to make a considerable programming shift and becoming a 80%-based Hip-Hop and R&B radio, a first in France history [Chakor, 2019] [Wikiwand, 2023e]. Indeed, up until that moment, radios have played very little of these two genres.

This, joined to the emergence of public television programs dedicated to this culture from the United States (such as "H.I.P H.O.P" [Wikiwand, 2023d]), will contribute to making Hip-Hop only more and more popular, spreading it across the whole country and inspiring the younger French generations to practice it.

All this artistic expression, passion and spreading will lead, in 2019, for Paris to be elected the most successful city for Hip-Hop by the American website [Oliver, 2020] based on record sales.

In the light of the information given in this section, this research thesis proposes trying to predict future music commercial success, which is a very strong indicator for the music industry and undoubtedly an objective pursued by all, and this research to take place in the French Hip-Hop market.

## 4.2 Research objective and detailed research question

My research objective is twofold:

1. What factors make a French Hip-Hop song successful?
2. How well can we predict the commercial success of a future French Hip-Hop song, based on the analyzed factors?

---

<sup>3</sup>This difference can be explained by the definition of "rapper" that we want to take here; indeed, some rappers could be linked to other music genres.

### 4.2.1 A correlation model

The first question will imply gathering data from commercially successful French rap albums in order to test factors, features that impact the success of its constituting songs. This will require creating a sustainable statistical model and then try to improve it based on the available scientific literature and eventually trade press.

### 4.2.2 A prediction model

Once this prediction model is created and running, we will go to step 2: trying essentially to predict new releases' commercial success. To simulate it, we are going to run our model on never prior observed data to check its accuracy.

Together, these two questions form my research question - **“Can the commercial success of a French rap song be predicted by features such as the context in which the said song is dropping out and its content?”**.

## 4.3 Envisaged method

The tools that I will use throughout this research thesis will be VSCode<sup>4</sup>, Python<sup>5</sup> and numerous of its libraries and MongoDB<sup>6</sup>.

### 4.3.1 Success definition and limitations

To achieve the above-stated objectives, we will first need to answer a simple yet essential question : “How can we define success?”; more exactly for us, what can we consider as a musical commercial success in the French rap industry? At this stage, a clear quantitative definition will be given and with it will also be defined its limitations.

### 4.3.2 Database creation and data to store

Once the definitions are given, we will carry on by creating our database and populating it with our data: album name, release year, artist name, and so on. This will be done thanks to programming. The database storing system will be MongoDB, as it allows to store semi-structured data, also called documents. This means that even if we encounter incomplete album data, the storing system will not break, and it will then be our job to replace it by some significant information if and when needed.

To this, among different features, we will also try to include some Natural Language Processing (NLP) analysis on the lyrics - retrieving the main covered topics of each song to have a clearer view on its content.

### 4.3.3 Hypotheses testing and model improvement

The features must be relevant - to know what criterias can be considered useful, we will rely on a relatively small set of features first, selected by intuition and review of success stories, and then improve it with two things, namely the results of our analyses and some literature review.

Also, it is worth noting that what we mean by ”success level” here is the official certification levels for the albums: we will analyze three of them, the most relevant ones,

---

<sup>4</sup>Visual Studio Code, also named VSCode, is a source-code editor made by Microsoft. [Microsoft, 2023]

<sup>5</sup>Python is a high-level, general purpose programming language. [Python, 2023]

<sup>6</sup>MongoDB is a document-oriented database management system that doesn't require specific data structure. [MongoDB, 2023]

namely Diamond, Platinum and Gold. These are also the classes that we will try to predict throughout this research thesis.

#### 4.3.4 Prediction model

At last, we will put in use our model by trying to predict future releases' commercial success level. Therefore, we will keep as an objective during this thesis to make our model reusable for further researches or analyses in our field.

All of our used code is available:

- Most important snippets: in the body of this thesis
- Relevant snippets: in the appendices
- All of our code: in our GitHub repository, available from [GitHub, 2023]

## 4.4 Outline

This research thesis proceeds in two main steps. Section 6 will dive into our work: starting with the theoretical framework, we will define which definition of "success" we have taken, taking the opportunity to point out its limitations. We then will give every information needed regarding methodology: the type of research, the data collection and analysis methods, and so on. Then, it will be time to dive into the scientific literature, mostly to gather new potential features to try. However, we obviously encountered some issues, and we will discuss these and the found solutions in the next section.

Once this is done, it will be time to present final results. We will observe the accuracy and robustness of our model, thanks to several different statistical and algorithmic techniques. This will be followed by some brief discussion.

The last step will be Section 7, where we will conclude. We will start by summarizing the limitations in regards to the hypotheses we have taken. At last, some of these limitations will be presented as new future research paths, suggesting to use our findings to pursue the analysis on this domain.

## Chapter 5

# Developments

## 5.1 Theoretical framework

### 5.1.1 Success: definition and limitations

To predict the success of a song, we first need to define what "success" is. [Webster, 2023], a public English dictionary, defines it as a "degree of measure of succeeding", a "favorable or desired outcome". So, re-phrasing our question, it becomes: what does it mean to be a commercially successful song in the music industry? As spoken in [Cross, 2016], although several metrics exist, there is one that is well-defined, objectively measurable and for which the data is abundantly available: music certifications.

Music certifications are granted by official organizations to reward artists and labels for meaningful achievements. To get these certifications, the artists and labels need to reach a certain level of sales, which is objectively determined by the organizations of each country.

In the United States - which is the leading country in terms of music revenues in 2022 according to [Smirke, 2023]-, the organization responsible for these awards is the RIAA<sup>1</sup>. RIAA was initially launched in 1958 and has known changes in certification conditions, notably due to streaming services arrival. Currently, the three main levels are as follows:

1. Diamond album: 10,000,000 units
2. Platinum album: 1,000,000 units
3. Gold album: 500,000 units

Additionally, multi-platinum albums are albums sold at at least 2,000,000 units, and they increment by 1,000,000 (a 3-times platinum albums is, therefore, an album that reached 3,000,000 units sold).

But since 2016, the international music industry has faced a big change. Indeed, in order to take into account the emergence of streaming platforms, which transformed our way of consuming entertainment (including TV programs, series and also music), **sales equivalences** have been introduced; starting from then, in the United States, a unit was equal to:

1. One digital or physical album sold or shipped
2. 10 tracks from a downloaded album
3. 1,500 on-demand audio or video streams of songs from the album

Now, for France, the organization that rewards certifications is called "SNEP"<sup>2</sup>, and its role is to be "the main employers' organization for producers, publishers and distributors of recorded music, partners of music artists in France" [SNEP, 2023b].

Its actual certification levels are as follows, according to [Wikiwand, 2023c]:

1. Diamond album: 500,000 sales equivalents
2. Platinum album: 100,000 sales equivalents
3. Gold album: 50,000 sales equivalents

---

<sup>1</sup>"RIAA" stands for Recording Industry Association of America. [Wikiwand, 2023b]

<sup>2</sup>"SNEP" stands for Syndical National de l'Édition Phonographique. [Wikiwand, 2019]

As for the United States, multi-platinum albums can be reached but by increments of 100,000 units, and up to 300,000 sales only. We observe that in France too, we take into account streaming through sales equivalence<sup>3</sup>. Here is the formula for unit sales equivalents:

$$sales\_equivalence = album\_sales + streaming\_sales \quad (5.1)$$

where

$$streaming\_sales = \frac{total\_album\_streams - \frac{most\_streamed\_album\_song}{2}}{1500} \quad (5.2)$$

Therefore, throughout this research thesis, we will always refer to this meaning of success - **a song that is commercially successful is a song whose album has received certification** (and obviously, the highest sales, the rarest certification, the better) - as it is objective, measurable and official.

It is also very important to note that another path was conceivable, namely focusing not on the album certification, but on the single certification. This category also exists and has produced sufficient amount of data<sup>4</sup>. However, we have decided to prefer focusing on albums for several reasons: first and foremost, we tried to explain success by the release context as much as possible, and we quickly reached the understanding that album releases have more information than single releases. Second, a single is oftentimes announcing an album release, which is in turn the real commercial target, as artists tend to make the most out of them, to capitalise on it with international tours, more promotional efforts, and so on. However, single certifications could have worked in pair with album certifications if we had decided to cross-check both; this will be discussed in the conclusions<sup>5</sup>.

## 5.2 Methodology

### 5.2.1 Type of research

The produced knowledge was basic, in the sense that it aimed **to develop an understanding on main features to predict the success of an album song**, while also being **applied**, as it provided a program to test the features and predict success.

Therefore, it surely was exploratory in the features' side of things, but it was **mainly explanatory**, as the most important thing to retrieve from this research was the model and its results.

**The approach was hybrid**: first, we tested a small sets of features, which is inductive approach, but then we analyzed the literature to test theories, which is deductive.

---

<sup>3</sup>The terms "sales equivalence" seem to have changed signification throughout time. We observe that at first in France, one spoke about sales equivalence when addressing the sales originating from streaming platforms exclusively but nowadays, it seems like more and more use these words to designate the total quantity of sales, physical and downloads included. We will prefer this second definition, as it seems more contemporary.

<sup>4</sup>For instance, to find the most recent certified singles in France: [SNEP, 2023a]

<sup>5</sup>What can be already said is we initially planned to work on the albums to correlate songs data inside of the latter to understand more accurately the reasons for commercial success. As it will be explained, we unfortunately did not do that.

**The data was mainly secondary**, as all statistics regarding albums was retrieved either from Genius or the SNEP website.

Since we analysed lyrics content, which is quantitative, but also worked with quantitative data (such as the page view number), **our research methods were mixed**.

We controlled our variables, trying to determine the causes and the effects, which made it **an experimental research**.

**Our sampling method was a non-probability one**: although we wanted to predict success of songs not in our set, we could only apply our findings to the specific subjects of our research.

We also selected data at a single point in time , which made it **a cross-sectional study**.

At last, as we adapted our features and rules during our research, our **design was flexible**.

Regarding the training model, we try to predict a class variable, which made it a **classification model**<sup>6</sup>.

## 5.2.2 Data collection and analysis

In this section, we are going to detail the most important steps taken to reach our objective, which is to create a model which could predict what song will be certified and to which degree. But in order to do that, we had to rely on some data, and more specifically data of previously certified albums, as it will be the basis of our research. Once we will have this data, we will pre-process it, removing noise and extreme values, to then store the important features into a database to be able working from it directly. After that, we will train our model based on unseen data, test its efficiency and try improving it with additional features or corrections.

Regarding data collection, the latter was completely based on the SNEP website[SNEP, 2023a], which provided all records used afterwards.

However, we didn't want to retrieve all genres of records; we wanted specifically French Hip-Hop releases. So, we had to find a way to filter records. To do so, we proceeded in several steps.

### Retrieving artist names

The first one was downloading the JSON file that gathered the list of all French Hip-Hop records ever put in [Genius, 2023], which is the leading music lyrics website and which has a very strong community around especially French rap music. In order to be correct and precise, we needed one unique source that has the most exhaustive list of artists possible, and Genius being a renowned Hip-Hop website, it was the perfect candidate to do so. Moreover and very importantly for me, it had a Python library, called LyricsGenius<sup>7</sup>, which allowed manipulating Genius records with ease.

---

<sup>6</sup>A sample of statistical model comparison made early in this research can be found in the Appendices, p.80

<sup>7</sup>[Miller, 2023]

```

1 #To retrieve the JSON file with all the data
2 album = genius.search_album("Discographie du rap français", "Genius France")
3 album.save_lyrics()

```

Source code 5.1: Retrieving all artist names through Genius

The code hereinabove shows exactly how we managed to retrieve all French rap albums ever published on Genius. "Discographie du rap français" being an album-like page artificially put up by the Genius community to list all releases year by year starting from 1982, we simply used `save_lyrics()`, a function allowing us to store all the information regarding an album in a JSON file.

The screenshot displays the Genius page for the 1995 rap discography. The main content is organized by month, listing specific releases with their release dates and artist names. The sidebar on the right provides interactive options for users to engage with the content, such as transcribing lyrics or adding song information.

Month	Release
Janvier	-
Février	-
Mars	* 21/03 : Les Fabulous Trobadors - <i>Quel sera notre futur ? (EP)</i> * 28/03 : NTM - <i>Paris sous les bombes</i>
Avril	* 19/04 : Les Fabulous Trobadors - <i>Ma ville est le plus beau park</i>
Mai	- 03/05 : Fabe - <i>Befa surprend ses frères</i>
Juin	* 02/06 : Assassin - <i>Homicide volontaire</i> * 05/06 : Cool et sans Reproches - <i>Plus français que moi tu meurs</i> * 16/05 : Ménélik - <i>Phénoménélík</i>
Juillet	-
Août	* 12/08 : Mellowman - <i>La voie du mellow</i>
Septembre	* 13/09 : Cut Killer - <i>Mixtape n°10 : Freestyle</i>
Octobre	- 20/10 : Akhenaton/Chili - <i>Métèque et mat</i> - XX/10 : Les Sages Poètes de la rue - <i>Qu'est-ce qui fait marcher les sages</i>

Figure 5.1: Example of a Genius page within this album

Now having one unique JSON file that contained a very large amount of French Hip-Hop releases with artist and release names, we needed to build a list of artist names that we could then input into the SNEP website. This artist name list could then be saved in a separate file, to proceed to further cleaning.



```

1 import json
2 import regex as re
3
4 #List to store all French rap artists
5 artists = []
6
7 with open('Lyrics_DiscographieDuRapfrançais.json') as json_data:
8     data = json.load(json_data)
9     i=0
10    for element in data["tracks"]:
11        text = data["tracks"][i]["song"]["lyrics"]
12
13        #We split the document into yearly releases
14        doc = text.split("\n")
15        #We only keep artist names
16        for line in doc[1:]:
17            #To remove dates
18            entry = re.sub("\- (.*/.*)\ :|\* (.*/.*)\ :", " ", line)
19            #To remove noise before artist names
20            entry = re.match(".*? (.*)\ -|^(.*)\ -", entry)
21
22            #We store new names
23            if(entry):
24                new = entry.groups()[0]
25                if new is not None:
26                    if new not in artists:
27                        artists.append(new)
28                else:
29                    new = entry.groups()[1]
30                    if new not in artists:
31                        artists.append(new)
32
33            i+=1
34
35        #We sort the list in alphabetical order before saving it
36        sorted = sorted(artists)
37        # print("Liste d'artistes: ", artists)
38        file = open("/Users/haternel/Downloads/diamondCertificationDB/artists.txt","w")
39        for artist in sorted:
40            file.write(artist + "\n")

```

Source code 5.2: Constructing the artist name's list

```
artists.txt x
SNEP_files > artists.txt
1 75ème Session
2 Alpha Wann
3 Artiste multiples
4 Beny, Yacine 1.5, Majdon Co & Yazid Z
5 Dabs
6 Dee Nasty
7 Dr Kimble, Débill Sass, Gaiden & Tidess
8 Gaiden & Scarp
9 L'Hexaler & O.B.L.
10 Navy
11 $-Crew
12 $ouley
13 $tanlee
14 Artiste multiples
15 Kheops
16 Indifonkigene
17 Dee Nasty
18 Destroy Man & Jhonygo
19 Jayden
20 Kemmler
21 100 Blaze
22 10Kret
23 10ver
24 111knu
```

Figure 5.2: Our *artists.txt* file

Now that we had a list of artists as exhaustive as possible, we cleaned it a second time and put every name in uppercase to avoid issues related to case sensitivity in the ordering (fortunately, the SNEP website was case-insensitive):

```
1 file = open("/Users/haternel/Downloads/diamondCertificationDB/artists.txt","r")
2 whole = file.read()
3 artists = whole.split("\n")
4
5 #Erasing the empty first element in the list
6 del artists[-1]
7
8 artists = sorted(artists)
9
10 #Storing in artists_filtered.txt
11 ret = open("/Users/haternel/Downloads/diamondCertificationDB/artists_filtered.txt","w")
12 filtered = []
13 for artist in artists:
14     artist = artist.upper()
15     if artist not in filtered:
16         filtered.append(artist)
17         ret.write(artist + "\n")
```

Source code 5.3: Cleaning furthermore the artist name's list

```
≡ artists_filtered.txt ×
SNEP_files > ≡ artists_filtered.txt
1  $-CREW
2  $OULEY
3  $TANLEE
4  100 BLAZE
5  10KRET
6  10VER
7  111KNU
8  113
9  13 BLOCK
10 13 ORGANISÉ
11 13MINI
12 1863
13 1984
14 1995
15 1KIZITION
16 1LYRE0
17 1PLIKÉ140
18 1KORRUPTIBLES
19 1MINUTE2RAP
20 1SPIRE
21 2 BAL 2 NEG
22 2 SQUATT
23 20SYL
```

Figure 5.3: Our *artists.txt* file, filtered

Now, we had everything we needed to retrieve all the records related to our French rap artists.

### Retrieving list of certified albums

Now that we finally have a usable, satisfactory and near-exhaustive list of French rap artists, we dived into the retrieval of records through the SNEP website, using Selenium<sup>8</sup>, a Python library that aims, among other things, at reading website pages and interacting with the latter in a methodical and automatic way.

```
1 import time
2 from selenium import webdriver
3 from selenium.webdriver.common.keys import Keys
4 from selenium.webdriver.common.by import By
5 from selenium.webdriver.chrome.service import Service
6 from selenium.webdriver.support.wait import WebDriverWait
7 from selenium.webdriver.support import expected_conditions as EC
8
9 #Retrieving our artist names' list
10 file = open("/Users/haternel/Downloads/diamondCertificationDB/artists_filtered.txt", "r")
11 whole = file.read()
12 artists = whole.split("\n")
13 del artists[-1]
14
15 #We specify that we want the webdriver to be of the Chrome type
16 # driver = webdriver.Chrome()
```

---

<sup>8</sup>[Muthukadan, 2023]

```

17 chromeOptions = webdriver.ChromeOptions()
18
19 #We define which destination folder we want
20 prefs_or = {"download.default_directory" :
21 "/Users/haternel/Downloads/diamondCertificationDB/snep_query/or"}
22 prefs_plat = {"download.default_directory" :
23 "/Users/haternel/Downloads/diamondCertificationDB/snep_query/platine"}
24 prefs_diamant = {"download.default_directory" :
25 "/Users/haternel/Downloads/diamondCertificationDB/snep_query/diamant"}
26 prefs_a_filtre = {"download.default_directory" :
27 "/Users/haternel/Downloads/diamondCertificationDB/snep_query/a_filtre"}
28 chromeOptions.add_experimental_option("prefs",prefs_a_filtre)
29
30 #We firstly scrap the site to accept cookies
31 cookies = "https://snepmusique.com/les-certifications/"
32 driver.get(cookies)
33 time.sleep(3)
34 cookie = driver.find_element(By.ID,"cn-accept-cookie")
35 cookie.click()
36
37 for element in artists:
38
39     #We retrieve the right address
40     start = "https://snepmusique.com/les-certifications/?certification="
41
42     #We define parameters
43     ##Levels of certification
44     certifs_diam = "Diamant,Double%20Diamant,Triple%20Diamant,Quadruple%20Diamant"
45     certifs_plat = "Platine,Double%20Platine,Triple%20Platine"
46     certifs_or = "Or,Double%20Or"
47
48     ##The rest
49     format = "categorie=Albums"
50     artist = "interprete=" + element
51     query = start + certifs_diam + "&" + artist + "&" + format
52     driver.get(query)
53
54     #We let the page load
55     time.sleep(3)
56
57     try:
58         #We check if there is an absence of results
59         empty = driver.find_element(By.XPATH, ("//div/h2[contains(text(),
60 'Désolé, aucun résultat ne correspond à vos critères de sélection')]"))
61         print(artist, " has not been found on the webpage.\n")
62     except:
63         #We click on the CSV button to download if there is a result
64         right_div = driver.find_element(By.CSS_SELECTOR,
65 'a.btn_red.btn_print.icon-download')
66         right_div.click()
67         print(artist, " has been searched in the webpage.\n")
68         time.sleep(3)
69
70 driver.close()

```

## Source code 5.4: Retrieving certified albums through SNEP

The output of this code was CSV files where each row represented a certified album with the following information:

- Artist name
- Project name
- Editor or distributor
- Type of project
- Certification level
- Release date
- Certification date

To carry on, we kept only some of the variables: namely, the artist and project names, the certification level and the release date.

Until now, we have retrieved aimed rap artist names and all of their nationally certified releases. Now, we wanted to dig into each of their certified albums to gather all the information we could to store it in a database.

### Retrieving certified albums' content

Until now, we only had the names of the albums that were certified, not their content. The following snippet allowed us to retrieve JSON files for each certified album in order to get meaningful information, namely the lyrics of each song of the albums as well as the song names and featurings.

```
1 from lyricsgenius import Genius
2 import json
3 import csv
4 import regex as re
5 from datetime import datetime
6
7 '''
8 Here, we can select from:
9 diamond_path = 'snep_query/diamant/dump_diam.csv'
10 platinum_path = 'snep_query/platine/dump_plat.csv'
11 platinum_issues = 'snep_query/platine/plat_issues.csv'
12 gold_path = 'snep_query/or/dump_gold.csv'
13 gold_issues = 'snep_query/or/gold_issues.csv'
14
15 With the following syntax:
16 with open(xxx, 'r') as yyy:
17
18 Where:
19 - xxx: path to the desired component
20 - yyy: name of the desired component
21 '''
22
23 with open(diamond_path, 'r') as dump:
24     csv_dump = csv.reader(dump, delimiter=";")
```

```

25     for row in csv_dump:
26
27         #Information that is used along the whole release
28         artistName = row[0].lower()
29         releaseName = row[1].lower()
30         releaseDate = row[5]
31         releaseDate_cleaned = datetime.strptime(releaseDate,"%d/%m/%Y")
32         print("Searching on Genius for: " +
33               releaseName + " of " + artistName)
34
35         album_spotipy = genius.search_album(releaseName, artistName)
36         releaseName_json = album_spotipy.name.replace(",","").replace("'", "").
37         replace(":", "").replace(" ", "").replace("#", "").replace("&", "").
38         replace("-", "").replace("=", "").replace(".", "").replace("(", "").
39         replace(")", "").replace("%", "").replace("$", "").replace("!", "").
40         replace("?", "").replace("-", " ").replace("(", "").replace(")", "").
41         replace("+", "").replace("'", "'")
42         releaseName_final = album_spotipy.name
43
44         album_spotipy.save_lyrics()
45         print("Album found on Genius")

```

Source code 5.5: Retrieving album's content

In this snippet example, we see that we take the diamond albums and we retrieve useful information, such as the artist name, the release name and date (which we format to standard).

There are two important steps left: first, we search the album on Genius based on two parameters: the release name and the artist name by using LyricsGenius' library. Once we have it, we clean the release name so as to avoid any issue later on: indeed, this name will have to match with the formatted name which is automatically given by Genius when we save the album information in JSON format. That is why we have this relatively consequent cleaning, which may not be optimal indeed.

The second important step left is to use the *save\_lyrics()* command from LyricsGenius to download the corresponding JSON file.

### Pre-processing album data

Now, we have the JSON files of each certified album we want. What is left to do is to pre-process this data and then store it into MongoDB. We do the following program record per record:

```

1  import json
2  import regex as re
3  from string import digits
4  from frenchLDA import topic_model
5
6  with open('Lyrics_' + releaseName_json + '.json') as f:
7      output = json.load(f)
8      i=0
9      instrumental = False
10     for song in output["tracks"]:

```

```

11     if output["tracks"][i]["number"] is None:
12         continue
13     featurings = []
14     themes = []
15     title = output["tracks"][i]["song"]["title"]
16     primary_artist = output["tracks"][i]["song"]["primary_artist"]["name"]
17
18     featured = output["tracks"][i]["song"]["featured_artists"]
19     song = output["tracks"][i]["song"]
20
21     if featured is None or len(featured) == 0:
22         featurings = "NULL"
23     else:
24         j=0
25         for features in featured:
26             featurings.append(output["tracks"]
27                 [i]["song"]["featured_artists"][j]
28                 ["name"])
29             j+=1
30
31     text = re.sub('^(.*\n){1}','',output["tracks"][i]["song"]["lyrics"])
32     if not text:
33         instrumental = True
34     else:
35         text = re.sub('You might also like','',text)
36         text = re.sub('Embed','',text)
37         text = re.sub('\[.*\]','',text)
38         text = re.sub(re.escape(releaseName) + '?.*\n','',text)
39         remove_digits = str.maketrans('', '', digits)
40         text = text.translate(remove_digits)
41
42     if not "stats" in output["tracks"][i]["song"] or
43         len(output["tracks"][i]["song"]["stats"]) == 0
44         or not "pageviews" in output["tracks"][i]["song"]["stats"]:
45         pageview = 0
46     else:
47         pageview = output["tracks"][i]["song"]["stats"]["pageviews"]
48
49     if instrumental or len(text.split()) <= 15:
50         themes = ""
51     else:
52         themes = topic_model(text)

```

Source code 5.6: Pre-processing data before storing

Once again, the pre-processing and cleaning part here is quite heavy. We will break it down into smaller pieces.

1. We make a loop on each song of the JSON file
2. We check if there is any data linked to the current element
  - (a) If the answer is yes: we carry on normally
  - (b) Otherwise: we skip this song (might be album cover, booklets, etc.)
3. We check if there is a featuring on the song

- (a) If the answer is yes: we mark '1' in the *featurings* variable
  - (b) Otherwise: we mark '0'
4. We check if there is some textual content in the current song
- (a) If the answer is yes: we remove the noise such as non-desired Genius promotional text or numbers
  - (b) Otherwise: we mark the boolean variable *instrumental* as True
5. We check if we have page views for the song
- (a) If the answer is yes: we write it in the *pageview* variable
  - (b) Otherwise: we write '0' in it
6. Finally, we check if we have enough text to analyze
- (a) If the answer is yes: we run the cleaned text in the *topic\_model()* function
  - (b) Otherwise: we let *themes* variable be blank

Before proceeding further, we are going to see what happens in the *topic\_model()* function.

## Topic modelling

When there was enough textual content to be analyzed (we put the threshold at 15 words, as we have observed with trials-and-errors that the different natural language processing (NLP) algorithms we used were able to return information above this limit), we used *topic\_model()* function, a function we created, which is as follows:

```

1  #Libraries
2  ##Text management
3  import spacy
4
5  ##LDA topic modelling
6  from gensim.models import Phrases
7  from gensim.corpora import Dictionary
8  from gensim.models import LdaModel
9
10 ##Storage
11 import json
12
13 #Launching the Spacy's library
14 stop_words = set(stopwords.words("french"))
15 nlp = spacy.load("fr_core_news_sm")
16
17 #LDA algorithm
18 def topic_model(text,artist,releaseName,songName):
19     ##Pre-processing
20     spacy_docs = nlp(text)
21
22     docs = []
23     for token in spacy_docs:
24         tokens = []
25         if len(token.orth_) > 2 and not token.is_stop:
26             tokens.append( token.lemma_.lower() )
27         docs.append(tokens)
28

```



```

29     ##Bigrams management
30     bigram = Phrases(docs, min_count=10)
31     for index in range(len(docs)):
32         for token in bigram[docs[index]]:
33             if '_' in token:
34                 docs[index].append(token)
35
36     ##Creating a dictionary with all the words
37     dictionary = Dictionary(docs)
38
39     ##Removing extreme values in the dictionary
40     if(len(spacy_docs) >8):
41         ''v2: dictionary.filter_extremes(no_below=1, no_above=0.25)''
42         ''v8: dictionary.filter_extremes(no_below=2, no_above=0.01)''
43         dictionary.filter_extremes(no_below=1, no_above=0.01)

```

Source code 5.7: Latent-Dirichlet Allocation (LDA) topic modelling

To understand this algorithm, we will proceed in several steps. Please keep in mind that each song having 15 words or more passes in this function one at a time.

In this first part, we take our lyrics and we convert them into a SpaCy’s list of words, called here ”documents”<sup>9</sup>. We then check that each document is at least 2 letters long and is not a stop-word (in NLP, one important thing to do is to remove noisy words, meaning frequent small words that form our sentences but don’t give much information, such as ”to”, ”at”, etc. These stop-words are defined per languages; here, working on French songs, we chose to load the SpaCy French stop-word list).

Once this is done, we take the lemma of the word we are actually considering, which is the standard, short version of the word, as to erase any variation (e.g.: ”lovers” becomes ”love”, ”eating” becomes ”eat”, etc) and to group together same topics. We then make sure the word is in small letters, and we fill in our list of tokens.

Once we have pre-processed our words, we can now create our dictionary using the function *Dictionary()*. One last concern are extreme values - what if we have very frequent words that can damage our topic modelling? To avoid that, we use the *filter\_extremes()* function. After testing several parameters combination, we reach a satisfactory result when we use *no\_below* = 1 and *no\_above* = 0.01, which designate the range of words we want to keep, and these are respectively present at least once in our dictionary and that compose no more than 1% of the latter.

Having defined our dictionary, we now launched our LDA model<sup>10</sup>:

```

1     ##Running the LDA model
2     corpus = [dictionary.doc2bow(doc) for doc in docs]
3     model = LdaModel(corpus=corpus, id2word=dictionary, num_topics=3,
4         chunksize=1000, passes=5, random_state=1, alpha='asymmetric')
5

```

<sup>9</sup>SpaCy is a Natural Language Processing (NLP) tool in Python. [Spacy, 2023]

<sup>10</sup>In natural language processing, Latent Dirichlet Allocation (LDA) is a Bayesian network (and, therefore, a generative statistical model) that explains a set of observations through unobserved groups, and each group explains why some parts of the data are similar. The LDA is an example of a Bayesian topic model. [Wikiwand, 2023a]

```

6      ##Storing in variables list of 5 words composing each topic
7      word_iteration = 0
8      for (topic, words) in model.print_topics(num_words=5):
9          if word_iteration == 0 :
10             words_1 = words
11             word_iteration += 1
12             continue
13         if word_iteration == 1 :
14             words_2 = words
15             word_iteration += 1
16             continue
17         if word_iteration == 2 :
18             words_3 = words
19             word_iteration += 1
20             continue

```

Source code 5.8: Running our LDA algorithm and storing results in variables

Here, we decided to take the next parameters:

- Our corpus is simply the words in our dictionary;
- We want to retrieve **3 topics per song**, which leaves room for discovering several subjects but doesn't force too much repetition for smaller texts;
- Each topic will have **5 words** constituting it, once again in the objective of having a compromise between not too many words (could fail for shorter texts) and not too little (allowing to make connections between several outputs)

```

{
  {
    "artist_name": "gims",
    "release_name": "Ceinture noire",
    "song_name": "Tant pis",
    "song": [
      {
        "topic_1": {
          "topic": null,
          "words": "0.034*\\"rien\\" + 0.034*\\"devoir\\" + 0.026*\\"faire\\" + 0.026*\\"prouver\\" + 0.026*\\"battre\\"""
        },
        "topic_2": {
          "topic": null,
          "words": "0.103*\\"homme\\" + 0.102*\\"trop\\" + 0.071*\\"donner\\" + 0.041*\\"douleur\\" + 0.040*\\"âm\\"""
        },
        "topic_3": {
          "topic": null,
          "words": "0.016*\\"dos\\" + 0.016*\\"agir\\" + 0.016*\\"devoir\\" + 0.016*\\"préférer\\" + 0.016*\\"rien\\"""
        }
      }
    ]
  },
  {
    "artist_name": "gims",
    "release_name": "Ceinture noire",
    "song_name": "Caméléon",
    "song": [
      {
        "topic_1": {
          "topic": null,
          "words": "0.055*\\"\\n\\n\\n\\" + 0.055*\\"laissé\\" + 0.055*\\"promettre\\" + 0.048*\\"passé\\" + 0.048*\\"année\\"""
        },
        "topic_2": {
          "topic": null,
          "words": "0.055*\\"rendez-vous\\" + 0.055*\\"danse\\" + 0.054*\\"vivre\\" + 0.054*\\"insister\\" + 0.016*\\"oui\\"""
        },
        "topic_3": {
          "topic": null,
          "words": "0.018*\\"promettre\\" + 0.018*\\"laissé\\" + 0.018*\\"\\n\\n\\n\\" + 0.018*\\"passé\\" + 0.018*\\"devant\\"""
        }
      }
    ]
  }
}

```

Figure 5.4: Preview of the Diamond certified releases after LDA model running

Having done that, we now needed to store the results in JSON files to re-use them afterwards.

```
1  ##Structuring output data
2  release_data = {
3      "artist_name": artist,
4      "release_name": releaseName,
5      "song_name": songName,
6      "song": [
7          {
8              "topic_1": {
9                  "topic": None,
10                 "words": words_1
11             },
12             "topic_2": {
13                 "topic": None,
14                 "words": words_2
15             },
16             "topic_3": {
17                 "topic": None,
18                 "words": words_3
19             }
20         }
21     ]
22 }
23
24 ##Storing in a JSON file
25 outfile = open("release.json", "r+", encoding="utf8")
26 outfile.seek(0,2)
27 json_object = json.dumps(release_data, indent=4, ensure_ascii=False)
28 outfile.write(json_object + ",\n")
```

Source code 5.9: Storing topic modelling result in JSON file

This design was thought to allow automation, as this particular structure enabled very easily navigating in the file and do the unique thing our LDA model could not do, which is automatically put a topic label for each suite of words. This label was therefore written by hand, replacing each "None" by the most coherent topic summarizing word. This will be discussed more in depth afterwards.

## Data storing in MongoDB

Now that we had all of the relevant data available and correctly formatted, we dumped our tracks information into MongoDB thanks to PyMongo. In this example, we are working with the Diamond collection:

```
1  from pymongo import MongoClient
2
3  # Initializing MongoDB client
4  client = MongoClient()
```

```

5 db = client["certificationDB_test_final"]
6 diamondcertificationDB = db["diamondcertificationDB"]
7 platinumcertificationDB = db["platinumcertificationDB"]
8 goldcertificationDB = db["goldcertificationDB"]
9
10 item = {
11     "artist_name" : artistName,
12     "release_name" : releaseName_final,
13     "release_date" : releaseDate_cleaned,
14     "certified_date" : certifiedDate_cleaned,
15     "track_number" : i+1,
16     "track_name" : title,
17     "primary_artist" : primary_artist,
18     "featured_artists" : featurings,
19     "track_lyrics" : text,
20     "themes" : themes,
21     "genius_pageview" : pageview
22 }
23 certificationDB.diamondcertificationDB.insert_one(item)
24 i += 1
25 print("Album stored in MongoDB")

```

Source code 5.10: Storing album data in MongoDB

## Creating a DataFrame in Python

Now that we had stored all of our albums in MongoDB, we could work with them more easily. The first thing to do was to load this data into Python and to do so, we needed a library to display databases and having functions to manipulate them at our convenience. So, we decided to use Pandas<sup>11</sup>, a broadly used library for these two targets.

```

1  ''' STEP 1/ LAUNCH '''
2  # Configure libraries
3  warnings.filterwarnings('ignore')
4  plt.rcParams['figure.figsize'] = (10, 10)
5  plt.style.use('seaborn')
6  pd.set_option('display.max_columns', 500)
7
8  # Initializing MongoDB client
9  client = MongoClient()
10 db = client["certificationDB_test_final"]
11 diamondcertificationDB = db["diamondcertificationDB"]
12 platinumcertificationDB = db["platinumcertificationDB"]
13 goldcertificationDB = db["goldcertificationDB"]
14
15 # Retrieving each collection
16 diamond_db=diamondcertificationDB.find()
17 plat_db=platinumcertificationDB.find()
18 gold_db=goldcertificationDB.find()
19
20 ''' STEP 2/ WORKING WITH PANDAS '''

```

<sup>11</sup>Pandas is a Python data analysis library. [Pandas, 2023]

```

21 # Constructing Pandas' DataFrame
22 df_diamond = pd.DataFrame(list(diamond_db))
23 df_platinum = pd.DataFrame(list(plat_db))
24 df_gold = pd.DataFrame(list(gold_db))
25
26 frames = [df_diamond,df_platinum,df_gold]
27 df_full = pd.concat(frames)

```

Source code 5.11: Creating Pandas' DataFrame

Since we were working with a considerable amount of data, we needed to make sure that we would be able to capture the most information possible, hence the maximum amount of columns displayed set at 500 for Pandas. The rest of the code above is very easily understandable, as it was just initializing the database, retrieving collections and constructing the DataFrame.

The concatenation of frames was aimed at merging collections together, as this was not the case by default.

```

1 # Dropping irrelevant columns
2 '''
3 Assumptions for now:
4 - artist_name: irrelevant, as we have the primary_artist field
5 - _id,track_lyrics: irrelevant
6 - certified_date: we want to avoid playing with durations
7 - release_name,track_name,track_number,primary_artist: not major factors, might be
8 tested later -> make other iterations of the model
9
10 Track_name is not erased here, since we need this condition checker
11 in array_dummifier_topics (first version). We erase it afterwards.
12 '''
13
14 df_full = df_full.drop(['artist_name','_id'
15                        , 'track_lyrics','certified_date'
16                        , 'track_number','release_name','primary_artist'
17                        ],axis=1)
18
19 # Missing values
20 '''print("Null values counter: \n",df_full.isnull().sum())'''
21
22 # Converting date variables into numbers before scaling
23 df_full['release_date'] = pd.to_datetime(df_full['release_date'])
24
25 # Extracting year from release_date
26 df_full['release_date'] = df_full['release_date'].apply(lambda x: x.year)
27
28 # Scale numeric data
29 df_full_copy = df_full.copy()
30 scaler = StandardScaler()
31 num_cols = ['genius_pageview','release_date']
32 df_full_copy[num_cols] = scaler.fit_transform(df_full[num_cols])
33
34 ''' STEP 3/ OUR FUNCTIONS '''
35 # Separating array cells before encoding

```

```

36 ## Function to explode, dummify and add prefix to categorical array variables
37 def array_dummifier_topics2(data):
38     # Extraction of unique topics
39     topics = set()
40     for col in ['topic_1', 'topic_2', 'topic_3']:
41         topics.update(data[col].apply(lambda x: x['topic'] if
42             isinstance(x, dict) else '').unique())
43
44     # Removal of null or None values in topics
45     topics.discard(None)
46     topics.discard('null')
47
48     # Creation of binary columns for each topic
49     for topic in topics:
50         data['topic_' + topic] = data.apply(lambda row: 1 if
51             any(row[col]['topic'] == topic for col in
52                 ['topic_1', 'topic_2', 'topic_3'])
53             if isinstance(row[col], dict) else 0, axis=1)
54
55     # Suppression of topic_1, topic_2, topic_3 columns
56     df = data.drop(['topic_1', 'topic_2', 'topic_3'], axis=1)
57
58     return df
59
60 ''' STEP 5/ CATEGORICAL DATA '''
61 # Dummifying topics
62 df_full_copy = array_dummifier_topics2(df_full_copy)
63 df_full_copy = df_full_copy.drop("track_name",axis=1)

```

Source code 5.12: Pre-processing of the DataFrame's data

Once again, we needed to clean and transform our data to fit our objective. Here, we followed this tutorial [Atha, 2021] and adapted where needed the following simple steps:

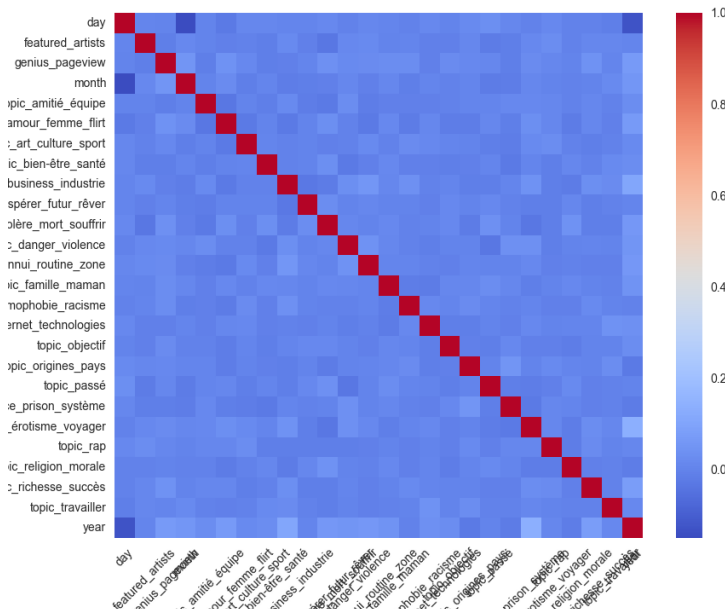
1. **Dropping irrelevant columns:** initially, only themes, featuring, page views and release data interested us, so we erased from the DataFrame superficial elements. Although track name was judged irrelevant for our analysis, we still maintained this column so far for the sake of further verifications.
2. **Checking for missing or odd values:** it was very important to make sure we have not got any weird values or errors in our database, as it would impact our analysis.
3. **Converting variables into the correct format:** here, only the date variable was changed, but it remains a very important check-up to make to use the right variables in the right format.
4. **Scaling numeric data:** using numeric values without a scale doesn't make sense; therefore, we need to pass our integers and floats into a standard scaler to even our numbers.
5. **Transform our topic columns:** indeed, until now we kept our topic columns in form of an object with two strings inside of it. However, to analyze our topics, here we can just retrieve the topic label. After that, we transform the labels into binary variables so as to compare records and their respective topics. At last, we delete the columns *topic\_1*, *topic\_2* and *topic\_3*, as we will not use them anymore.

## Checking variable independence

Once the model is constituted, the next big step is to check whether there are dependence issues to arise or not. To do so, there are several different techniques, but the one we used first from the topic modeling step (see 5.26) is called the **heatmap**.

```
1 ''' STEP 6/ CHECKING VARIABLE INDEPENDENCE '''
2 correlation_matrix = df_full_copy.corr()
3 plt.figure(figsize=(10, 8))
4 sns.heatmap(data=correlation_matrix, annot=False, cmap='coolwarm')
5 plt.xticks(rotation=45)
6 plt.yticks(rotation=0)
7 plt.show()
```

Source code 5.13: Defining our heatmap



Source code 5.14: Results of our heatmap

This graph represents the evidences of correlation between our different features. On both axes, we find the same features; this is normal, as the intention is to compare each pairs of features together. This also explains why we see a diagonal red line: it is the identity pair.

As one can observe, our model seems no to suffer from variables dependence, which is good<sup>12</sup>.

## Training our model

Once our DataFrame was finally in the format we desired, it was time to train our model. We decided to take the double-split method in order to divide our dataset; although other methods exist, this one worked just fine and was very easy to understand. After trying several combinations, the final split was made as such:

<sup>12</sup>For an insight on encountered difficulties regarding heatmap creation, please refer to 5.2.4

- Training set size: 70% of the dataset
- Validation set size: 15% of the dataset
- Test set size: 15% of the dataset

This is what we observe in the following code snippet:

```

1  ''' STEP 7/ TRAINING THE MODEL '''
2  # Set training & testing data
3  ''' DOUBLE SPLIT METHOD'''
4  ## First split: train / other
5  X_train, X_rem, y_train, y_rem = train_test_split(feature, target,
6                                                    train_size=0.70)
7
8  ## Second split: validation / test
9  X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem,
10                                                     train_size=0.15)

```

Source code 5.15: Double-split method to train our model

Now, what model to chose? In order to make sure to take the right one, we thoroughly tested the following ones:

- Random Forest
- Decision Tree
- Naive Bayes
- Logistic regression
- K-nearest neighbors
- Support Vector Classifier

One can find the respective code snippet for each algorithm in the Appendices, starting from p. 70.

By using a **classification report** to help us evaluate the performance of the models and to give us an overview of the latter on the dataset, we were able to run our comparison tests much more rigorously and methodically. This produced table figures that one can found in the Appendices p. 71.

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,74	0,66	0,7	154
<b>Platinum</b>	0,88	0,82	0,85	941
<b>Gold</b>	0,8	0,87	0,83	976
<b>Accuracy</b>			0,83	2071
<b>Macro average</b>	0,81	0,78	0,79	2071
<b>Weighted average</b>	0,83	0,83	0,83	2071

Table 5.1: Classification report: oversampling RF + day/month/year variable split



In this section, we will only discuss the results of our final and most prediction-efficient model, which is the oversampling Random Forest version<sup>13</sup> that also benefits, in its feature variables, from the split between the release day, month and year.

First, let us take a quick look at each axis to have a good understanding of the results. Each classification report aims to calculate, among other things, how well can we predict each certification level (Diamond, Platinum and Gold). These predictions can be averaged one of the two following ways: either taking into account the proportion of each real instance in a given class (also named the "Support"), or not, giving us respectively weighted & macro averages for a given class.

These are our rows. Now, in the columns, we find our metrics:

- Accuracy: the total ratio of correct predictions on all predictions made
- Precision: ratio of correct predictions for a given class
- Recall: ratio of real instances of a given class correctly classified by the model
- F1-score: harmonic mean of the precision and the recall given a class

Now, we needed to decide which metric to optimize. Our goal was **to predict the correct certification as consistently as possible, no matter the certification**; therefore, we are less interested by the macro average than the weighted one, as the first does not take into account the disproportion in our data sample, meaning it will be influenced by the most dominant class and potentially lead to higher prediction results than in reality.

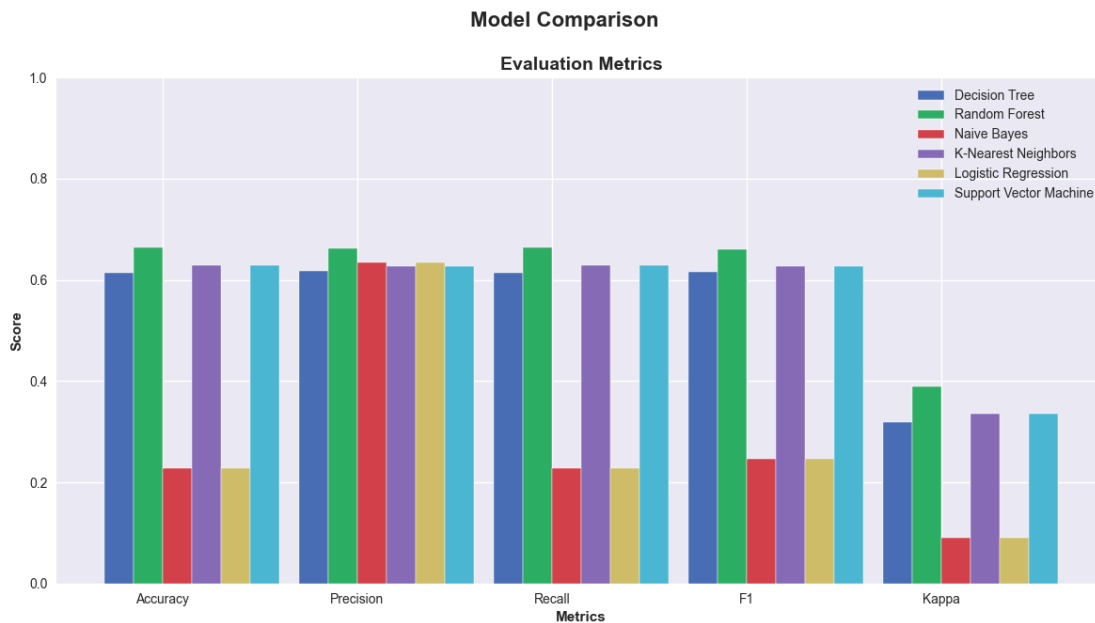


Figure 5.5: Model comparison without weighting average

Now, we have to pick between precision, recall, F1-score and accuracy. Precision and recall give us metrics per class, but we are more interested in the overall performance of our model. Fortunately, F1-score and accuracy do just that for us. At last, the accuracy can be a good metric to follow, but it is unfortunately impacted by the unweighted samples, too, which is not the case of the F1-score. The F1-score, indeed, allows us to capture

<sup>13</sup>Regarding the oversampling technique used, please refer to the next section, p. 40.

all the efficiency of our model, whether it is in terms of precision or accuracy, without forgetting the least present class in our sample. Therefore, **we choose to maximise the F1-score metric** in this research thesis.

Now, to come back to our classification report showed hereinabove, we indeed observe a major disproportion between the Diamond class, with only 154 instances, compared to Platinum and Gold (respectively 941 and 976 instances). However, the algorithm is able to run correctly, as the precision and the recall remain at sufficient level for Diamond releases, although one can notice a significant decrease in recall compared to the rest. Yet, the F1-score, once weighted and averaged, outputs very good prediction capabilities overall.

Therefore, and after running extensive trials on different training algorithms whose results can be observed in the Appendices p. 71, we found that **Random Forest has the highest F1-score in weighted average**, and therefore we will keep this algorithm for the rest of the research.

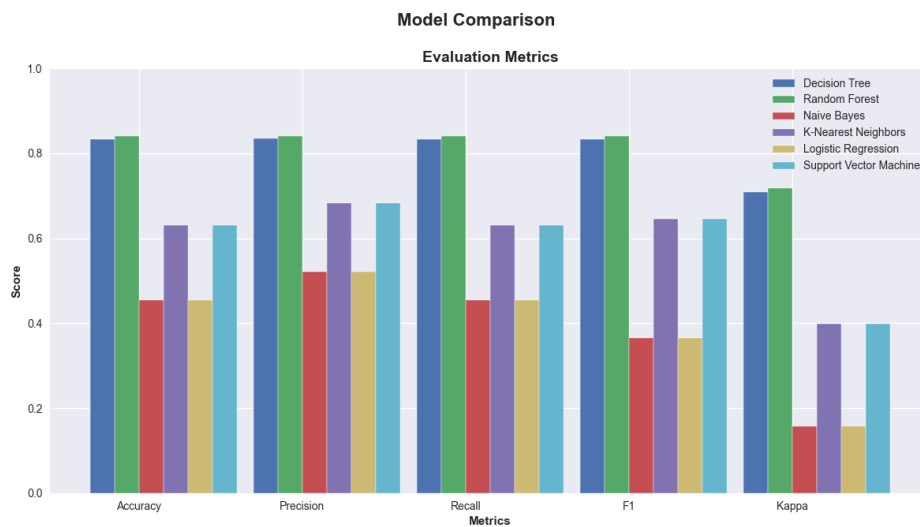


Figure 5.6: Model comparison with oversampling

## Improving our model

The next question was to know if we could improve our model and if so, how. As before, several approaches were used to try and reach a better result<sup>14</sup>:

- **Balancing the set:** here, we tried to fix the unbalance between the diamond set and the two others; indeed, we observed a considerable difference:
  - Diamond set: over 2,000 songs
  - Platinum set: over 7,000 songs
  - Gold set: over 7,000 songs

This unbalance could potentially explain the poor results obtained when our models tried to predict the diamond songs.

Therefore, we tried the following fixes:

- Data oversampling, using the *SMOTE()* function from the *imblearn* library
- Data undersampling, using the *RandomUnderSampler()* function of the same library

<sup>14</sup>For the corresponding code snippets, please refer to the Appendices.

- A combination of both previous approaches, in the form of the *SMOTTEEN()* function (still the same library)

In this set of solutions, it was the **hybrid SMOTTEEN method** which worked the best

- Searching for the best predictor model: testing each model was fastidious. It remained possible, but not efficient. This is why we tried to approach it methodically by using the *VotingClassifier()* function of the *sklearn* library, which allowed to compare several prediction models at the same time and produce the best output possible. In this set of tests, **Random Forest Tree** ended up still being the most relevant model.
- Searching for the best set of parameters for our model: another way of tweaking our results is to directly change parameters of the current model. Once again, this can be done manually or automatically. For efficiency's sake, we preferred the second option by using the function *GridSearchCV()* of *sklearn* library. Here were the different combinations of parameters that we tested
  - max\_depth: 50, 80 or 100
  - max\_features: 2, 3 or 4
  - min\_samples\_leaf: 3, 4 or 5
  - min\_samples\_split: 8, 10 or 12
  - n\_estimators: 100, 300, 500, 750 or 1000

The obtained results concluded that **no major improvement could be done**, as it returned very similar results to what we already had with the oversampled Random Forest approach.

- Modifying the set of features considered: what if some were useless, or on the contrary, what if new features could tremendously improve our model? This is what we tried to explore by running a feature importance analysis, specifically designed for classification models. It revealed to be really insightful, as it was able to highlight less or more important elements to work around.

This, in turn, informed us that **our model had relatively relevant features that could not be set aside**, as it would hinder our prediction abilities<sup>15</sup>.

After all these observations and testing, it was now time to try improving our model in a different manner. Until now, we have tried to change or, at best, remove elements from our model. By referring to existing scientific literature in place, we would be able to search for new features to take into account in the continuous effort of improving our predictions.

### 5.2.3 Scientific literature review

The aim of the section could be defined as building upon the model we currently have in order to improve our prediction abilities. But in order to improve it, we must consider new relevant values, and these can be found in the available scientific literature on this subject. Hereunder, we only consider the literature that serves our purpose by bringing the most value to our research.

The scientific literature we found trying to experiment and explain music success through some features could be divided into 5 different explanation topics: artist popularity,

---

<sup>15</sup>To observe related results, please see 5.9

marketing efforts, the time and the place, the post-release pieces of information and the scientific approach. We propose to look at it here afterwards.

1. **Artist popularity** We already lightly touched this subject by taking into consideration the lyrics page views number, which could to some extent represent the popularity of the artist. This, of course, is not the only way to measure the popularity of an artist, and is arguably not the most complete way of considering the latter. On that note, many researchers tried to encapsulate data which could define or impact artist popularity: among others, [Bhattacharjee et al., 2007] proposed, in his paper about music piracy and its impact on music success seen through the scope of music ranks, that the **superstar status** could have significant impact on artist success. What he referred to was the status that a very small percentage of artists have, a status where artist tickets, concerts and previous albums' sales reached massive numbers, numbers so big that the latter oftentimes entered the pop culture, and whose next project is expected with considerable "hype". And this could be interesting to inspect, as this suggestion was made back in 2007; since then, it is obvious to us that by having even more superstar examples in the industry, the research would only be more accurate. [Strobl and Tucker, 2000] extended on that topic, speaking of "**consumption capital**", when the artist becomes so popular and so renown that consuming its craft becomes mainstream and easy, as opposed to searching for and experiencing new artists' craft, which can be time-consuming and unsatisfactory.

However, that is not all. Artist popularity's status could also be considered through its **total number of sales** or its **number of certified releases**, [Lee et al., 2003] proposes. This is a more numeric approach that suggests taking advantage of the great number of available data online, which is what we tried to do in our research. Knowing that this study has been made in 2003, we think it would be very interesting to update it nowadays.

At last for this part, [Strobl and Tucker, 2000] & [Aguiar and Waldfogel, 2021] both spoke of popularity in terms of **social network and public engagement** importance, emphasizing the impact of proximity to one's audience and the network effect, where the more people listen to your music and share their preferences, the more new people join the trend. To take this approach, however, it would be necessary establishing threshold values that could be as clear and objective as possible to test this feature's influence. Moreover, it would be important to check whether the network effect can be at least partially responsible for mid- or long-term success, or rather if this effect fades out quickly after a sudden burst.

2. **Marketing efforts** In addition to artist's popularity, one can also consider **marketing techniques** to increase artist awareness and solidify its positioning or **music branding**, as suggested by [Berns and Moore, 2012]. Indeed, music can be seen as cultural good, therefore benefiting from usual marketing approaches. Several pieces of research, namely those of [Lee et al., 2003], [Bhattacharjee et al., 2007] & [Dertouzos, 2008] go even further, as they prove the importance of **getting on the radio** to boost up music sales. Still, it remains important to keep in mind that this research dates from 2008, allowing to ask oneself what impact radio presence plays in the current times. [Lee et al., 2003], for instance, speaks about the radio as a very necessary step to promote one's music, the latter's achievement therefore witnessing the **promotional efforts** put into the release.

[Aguiar and Waldfogel, 2021] even goes a step further by considering the impact of **getting on Spotify official playlists**, and this research proves indeed the very

fruitful consequences for the artist and its popularity as seen through its subscriptions and most evidently, its monthly streams. But here also, it would be important to verify whether this impact is a short-term trend or if this has longer term consequences. Another limitation to take into account here is that searchers only considered songs entering once in the said playlists, whereas some songs may enter several times, potentially re-surging after a first fall in the ranks.

3. **Time and place** [Bhattacharjee et al., 2007] further suggested two features regarding the context in which the project releases: its **year period release** and the **artist label**. The first is motivated by the simple observation of New Year’s holidays’ impact on consumption, where sales tend to boost up at approaching Christmas and the New Year Eve; the second is linked to the different work methods of labels, regarding whether one speaks about a major label, with obviously more funds and potentially a bigger network, or an independent label, with fewer of both.
4. **Post-release information** Until now, we have seen potentially important features that had in common their precedence to the project release, meaning it was information publicly available even before the project came out. [Bhattacharjee et al., 2007] suggests also exploring post-release information, such as **online reviews** on the project to further reach bigger audience through the public itself, or more precisely the network effect in action. These reviews could in fact intrigue more listeners to give the project a try or on the contrary, spare their time listening to it.

But by further diving into platforms & web available data, we also observe that **the start position in the charts**, also called "rank", could be an interesting indicator of later success, as it is also proposed by [Bhattacharjee et al., 2007]. Indeed, the searchers observed that projects tend to stay or decrease in ranking, but rarely do they improve their rank. Naturally, this observation could be counter-tested (or not) nowadays, as this study is 16-years old, but it still remains an interesting feature to test.

But what about music itself? That could, too, be success indicator. Features like **album & artist genres**, respectively suggested by [Strobl and Tucker, 2000] & [Lee et al., 2003], could prove be relevant, although those tend to blend more than ever before, asking the question of categorization relevance in this thesis. [Kellaris and Kent, 2023], on the other hand, developed interest in the **music tempo, tonality and texture**, and highlighted success prediction relevance. Therefore, and since each music genre as its own rhythm, tonality and texture, this somehow supports the suggestions of the previous research regarding album and artist genre relevance. At last, [Berns and Moore, 2012], in his paper about neural prediction of cultural success, emitted the hypothesis that **music quality** could play a prediction role, too.

5. **Scientific approach** Speaking of the research of [Berns and Moore, 2012], the latter focused on the **neuro-biological approach** to explain music tastes, hence music consumption. The study tested what areas of the brain reacted to newly discovered music, in link to listener’s rating of the said music. It turned out that this approach could also at least partially explain music success, although the author emitted questions about sample representativeness and objectivity of cerebral activation.

The outcome of the scientific literature review was to consider one additional feature in our model in the name of **the year time period in which the album was released**, although many of the ones stated hereinabove would undoubtedly bring interesting insights, due to a desire to focus on already-running model instead of trying potentially

time-consuming changes and improvements.

When done so, we re-ran our feature importance analysis and we indeed observed that this new data split was even better at predicting success than the previous one.

For the feature importance analysis source code, please refer to the Appendices p. 80.

File name	Features	Algorithm	Results
model.py	Themes Feats Release year Views	Random Forest	Feature: featured_artists, Importance: 0.027317254023263543 Feature: genius_pageview, Importance: 0.370468487027545 Feature: release_date, Importance: 0.2688691641994313 Feature: topic_amitié_équipe, Importance: 0.015784558851689078 Feature: topic_amour_femme_flirt, Importance: 0.023870541303234308 Feature: topic_art_culture_sport, Importance: 0.019348867959312704 Feature: topic_bien-être_santé, Importance: 0.016563662776867046 Feature: topic_business_industrie, Importance: 0.02271603334817243 Feature: topic_changer_espérer_futur_rêver, Importance: 0.014067612010333807 Feature: topic_colère_mort_souffrir, Importance: 0.02827176847813169 Feature: topic_danger_violence, Importance: 0.028582913790714884 Feature: topic_ennui_routine_zone, Importance: 0.021798255874058915 Feature: topic_famille_maman, Importance: 0.018664329988451644 Feature: topic_homophobie_racisme, Importance: 0.0039822059632673105 Feature: topic_internet_technologies, Importance: 0.00630996013538994 Feature: topic_objectif, Importance: 0.005959920725709025 Feature: topic_origines_pays, Importance: 0.009052252343777442 Feature: topic_passé, Importance: 0.014211655639050533 Feature: topic_police_prison_système, Importance: 0.014974038498649238 Feature: topic_profiter_érotisme_voyager, Importance: 0.031168882601733713 Feature: topic_rap, Importance: 0.006071679854171031 Feature: topic_religion_morale, Importance: 0.016417129944788958 Feature: topic_richesse_succès, Importance: 0.015325606929450918 Feature: topic_travailler, Importance: 0.00020321774616094212

Figure 5.7: Feature importance analysis of our first model version

File name	Features	Algorithm	Results
model2.py	Themes Release month Feats Views Release year	Random Forest	Feature: featured_artists, Importance: 0.023318267716635716 Feature: genius_pageview, Importance: 0.26175065486509475 Feature: month, Importance: 0.17226965620703108 Feature: topic_amitié_équipe, Importance: 0.013339947159455544 Feature: topic_amour_femme_flirt, Importance: 0.0240418250875914 Feature: topic_art_culture_sport, Importance: 0.015534959672319904 Feature: topic_bien-être_santé, Importance: 0.014065816429239762 Feature: topic_business_industrie, Importance: 0.022137124513367665 Feature: topic_changer_espérer_futur_rêver, Importance: 0.01344125163993136 Feature: topic_colère_mort_souffrir, Importance: 0.030132832479567058 Feature: topic_danger_violence, Importance: 0.02910442042803169 Feature: topic_ennui_routine_zone, Importance: 0.020985865524623035 Feature: topic_famille_maman, Importance: 0.01588730592140278 Feature: topic_homophobie_racisme, Importance: 0.0025762137277663875 Feature: topic_internet_technologies, Importance: 0.0046710422810901115 Feature: topic_objectif, Importance: 0.0032091691795146974 Feature: topic_origines_pays, Importance: 0.008023782992532008 Feature: topic_passé, Importance: 0.011166566520669738 Feature: topic_police_prison_système, Importance: 0.013580049128875962 Feature: topic_profiter_érotisme_voyager, Importance: 0.030541310869336254 Feature: topic_rap, Importance: 0.004474899167146002 Feature: topic_religion_morale, Importance: 0.015062012354993978 Feature: topic_richesse_succès, Importance: 0.014944811945423925 Feature: topic_travailler, Importance: 0.00020603637502281203 Feature: year, Importance: 0.2355341778133363

Figure 5.8: Feature importance of model version 2 - taking release month into account

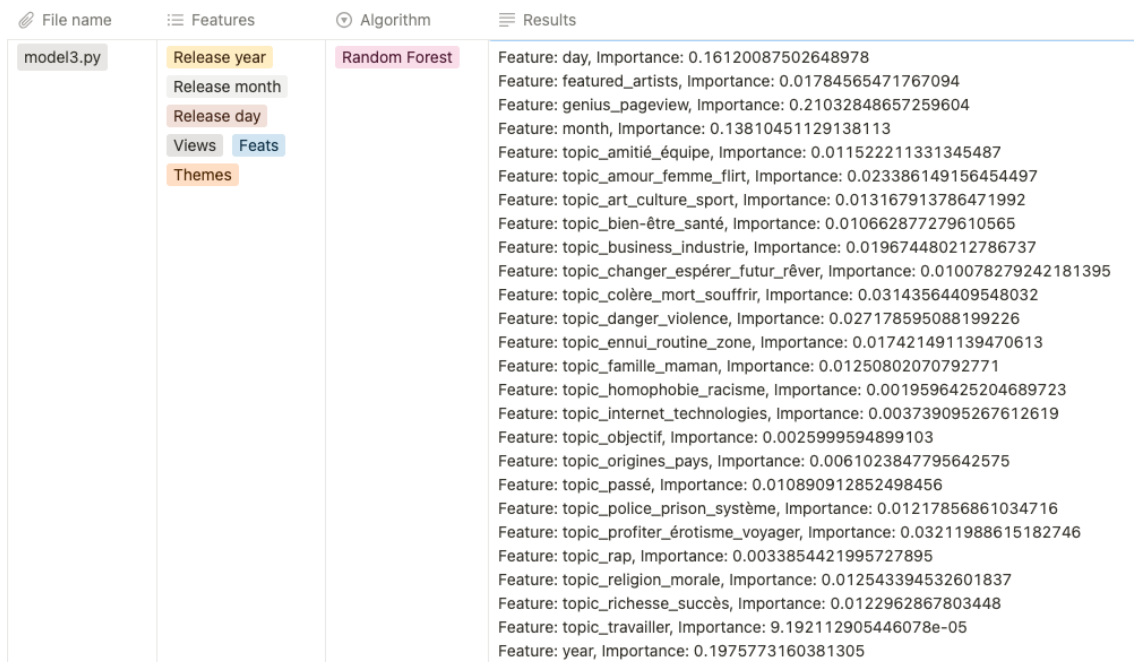


Figure 5.9: Feature importance of model version 3 - taking release day into account

One can observe hereinabove our different versions and their respective results. We already discussed the first version, with our initial features. The second explored the same features, with the addition of the release month. The last one added the release day. By adding the three date components together, we observe a better prediction ability than initially (reaching no less than **49.69% of feature importance** when we started with 26.87%), and that without losing too much value anywhere else, which is a strong improvement indeed.

## 5.2.4 Problems encountered and solutions found

This part is dedicated to the most significant issues we encountered during our research thesis. First, we will explain what the issue was: what caused it and what impact it had. Second, we will detail how we managed to work around it: what tests we made, what we tried to do, and essentially what was our solution to fix it. At last, we will briefly explain what impact such changes had.

In overall, our main model issue was **overfitting**, which can be described as having too strong or merely impossibly accurate predictions in our data, oftentimes consequential to a model which struggles to generalize but is good at getting the specificities of the currently observed information and hence, is making very strong results but only in our specific data. The related fixes found will be discussed in pp. 57 & 60.

### Creating a coherent heatmap

A heatmap, as explained before, is a graphical representation of the relation between variables in a specific DataFrame. The aim of the heatmap is to assess whether dependence issues may arise or not between variables, which would lead to erasing, replacing or in any case modifying our current model.

The issues we encountered with this tool is to put in parallel with issues in the fields of topic modelling and topics quantity in overall. Indeed, when we started running our first codes, we would observe very strange results, such as the following.

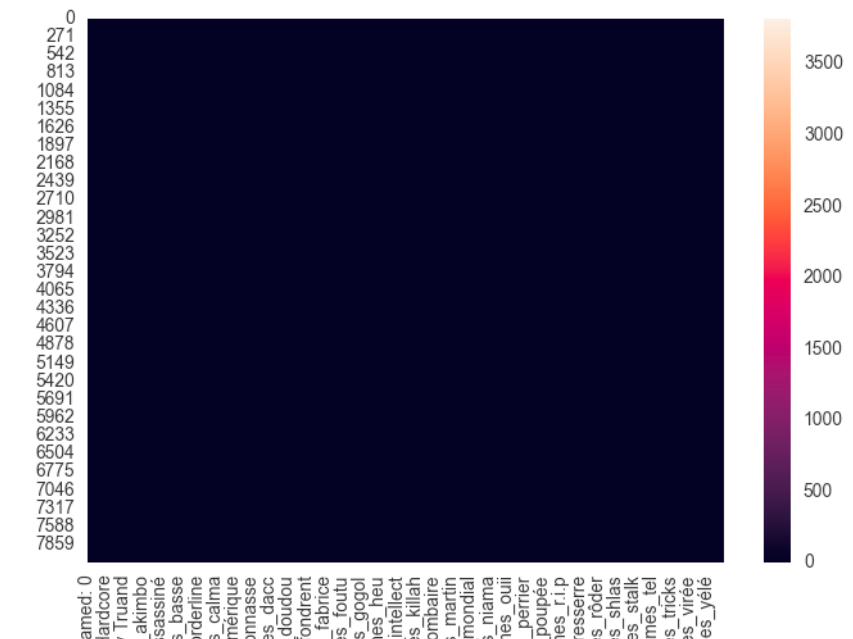


Figure 5.10: Our first heatmap result, which is completely incorrect

```

1 p1 = sns.heatmap(df_full_copy)
2 plt.show()

```

Source code 5.16: Our Python code for our incorrect first heatmap

What were the main issues here ? Well, everything:

1. Variables must be the same on both axes (as we try to assess dependencies and correlations), which is not the case here
2. On the Y axis, we find a series of number that does not make any sense
3. The heatmap is completely blank
4. There are too many variables taken into account at once
5. A heatmap must work starting from a correlation matrix, which is not given here

So we proceeded to try correcting the heatmap. Without it, we had less insight on what was going on in our model, which is unfortunately the objective of this graph.



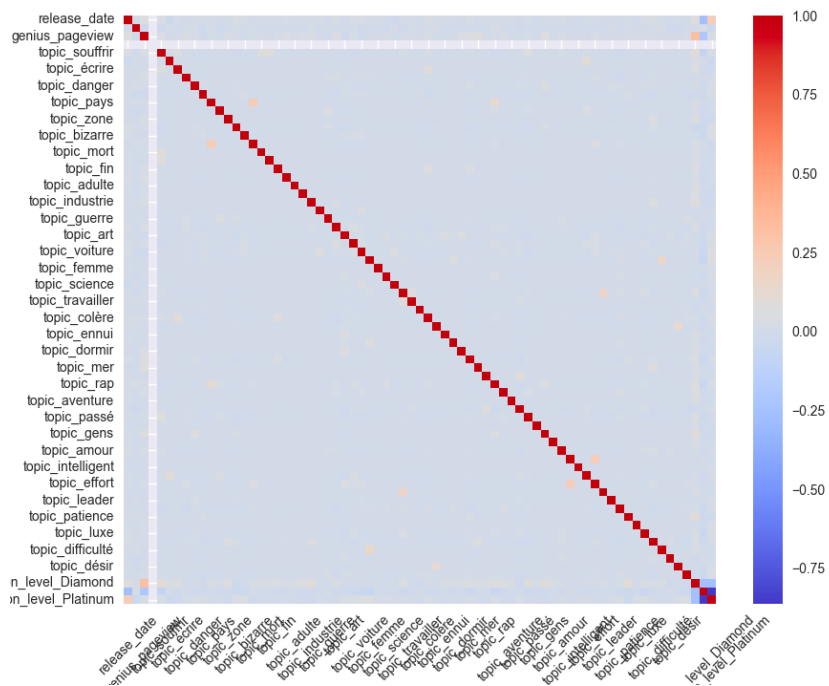


Figure 5.11: Our second heatmap, with correlation matrix

```

1 correlation_matrix = df_full_copy.corr()
2 plt.figure(figsize=(10, 8))
3 sns.heatmap(data=correlation_matrix, annot=True, cmap='coolwarm')
4 plt.xticks(rotation=45)
5 plt.yticks(rotation=0)
6 plt.show()

```

Source code 5.17: Our second heatmap

This heatmap was much better already. Here, we could observe no significant correlation between variables, although some negatively correlated features seemed to appear regarding certification levels. We were quick to remark that these variables had nothing to do here, as they were not part of the features analyzed.

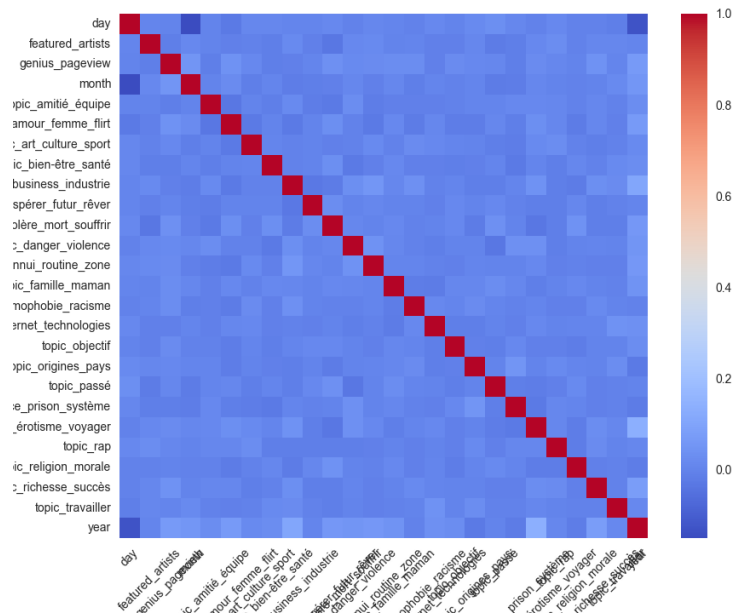


Figure 5.12: Heatmap, third version: less features, removal of certification levels

```

1 feature = df_full_copy.drop('certification_level',axis=1)
2 target = df_full_copy['certification_level']
3
4 correlation_matrix = feature.corr()
5 plt.figure(figsize=(10, 8))
6 sns.heatmap(data=correlation_matrix, annot=False, cmap='coolwarm')
7 plt.xticks(rotation=45)
8 plt.yticks(rotation=0)
9 plt.show()

```

Source code 5.18: Correct code for the heatmap

At last, we arrived at satisfactory results. Following issues were fixed:

- Less topic labels (in relation to topic modelling)
- Removal of certification levels as analyzed features
- Correction of axes
- Creation of a correlation matrix

Once we reached this result, we ran a k-fold cross-validation analysis on our accuracy to find that in that area too, overfitting-related issues were fixed<sup>16</sup>

### Choosing the right topic modelling algorithm

One of our main problems was undoubtedly topic modelling. We spent a lot of time in research of a satisfying topic model algorithm. Our difficulties can be summarized into

<sup>16</sup>One can see the result in the Appendices, p. 84. Right after it at p. 84, one can also see that the written output gave us "NaN" arrays for other metrics, which we unfortunately could not resolve at time, and thus although the model was running smoothly.

two different aspects: the research of the right algorithm and the research of the right parameters.

To find the right algorithm first, we had to review articles and make tests to see the best fit. Fundamentally, we tried two types of natural language processing algorithms: **word embedding algorithms** and **Latent Dirichlet Allocation (LDA) algorithms**. These algorithms were tried on two arbitrarily chosen albums: "PSLP" by Pollux<sup>17</sup> and "13 organisée" by 13 Organisé<sup>18</sup>. The choice was made such as to have, for the sake of representativity and extremes' testing, respectively:

- "PSLP" by Pollux: a project with **5 songs** which are on average **400 words** and **3 minutes long**, with **no presence of featurings**
- "13 Organisé" by 13 Organisé: a project with **14 songs** which are on average **1,000 words** and **5 minutes long**, with **a lot of featurings**

We initially struggled to write a working program with the LDA algorithm. Indeed, as one can see in Figure 5.19, we observed very repetitive patterns in our output (which were, moreover, not very insightful) when we ran the program on "PSLP" by Pollux, meaning the album which had not so much textual content. To some extent, the same problem could be seen with the second album, especially regarding repetition.

```
Topic1:0.017*"fond" + 0.017*"ouai" + 0.017*"loin" + 0.017*"trop" + 0.012*"main"  
Topic2:0.009*"loin" + 0.009*"ouai" + 0.009*"fond" + 0.008*"dis" + 0.008*"âme"  
Topic3:0.009*"trop" + 0.009*"ouai" + 0.009*"fond" + 0.009*"loin" + 0.009*"night"
```

Source code 5.19: Repetitive patterns for words in initial LDA version

That is why we turned our attention to word2vec, which seemed to work sufficiently enough to be interesting. However, word2vec does not work like LDA: whereas the latter returned us a sequence of words which represented a certain number of topics, in our program, word2vec returned us the 15 most frequent words, once the pre-processing and filtering were done. This is not the same as returning topics, but we unfortunately failed to observe that initially.

Therefore, we went on and ran our program with word2vec, whose code can be seen hereunder.

```
1 from gensim.models import Word2Vec  
2 import spacy  
3 import nltk  
4 from nltk.corpus import stopwords  
5 import re  
6 from gensim.models import Phrases  
7 from gensim.corpora import Dictionary  
8 from sklearn.decomposition import PCA  
9 from matplotlib import pyplot  
10 from gensim.test.utils import datapath  
11 from gensim import utils  
12 import gensim.models
```

---

<sup>17</sup>[Genius, 2022]

<sup>18</sup>[Genius, 2020]

```

13
14 def embedding(doc):
15     model = Word2Vec(doc)
16     stop_words = set(stopwords.words("french"))
17     nlp = spacy.load("fr_core_news_sm")
18     wholeText = re.sub("\\n", " ", doc)
19
20     spacy_docs = list(nlp(wholeText))
21
22     docs = []
23     for token in spacy_docs:
24         tokens = []
25         if len(token.orth_) > 2 and not token.is_stop:
26             tokens.append(token.lemma_.lower())
27             docs.append(tokens)
28
29     model1 = gensim.models.Word2Vec(docs, min_count = 1, workers=4, sg=0)
30     return model1.wv.index_to_key[:15]

```

Source code 5.20: Our program for word2vec topic modelling

This program is very similar to the one we are using in our current model, except when we run the topic model.

As we said, we ended up with a list of 15 words, ranked from the most frequently appearing one to the least one. This is what we initially put into our storage system, in a column called *themes*.

Changing that required accessing to the storage, retrieving the right columns (called "fields" in MongoDB), re-running our topic modelling algorithm, labeling the said topics and storing in the right place the new data in new fields. And this is precisely what we can see hereunder:

```

1  # Importing libraries
2  from pymongo import MongoClient
3  import pandas as pd
4  import json
5
6  # Initializing MongoDB client
7  client = MongoClient()
8  db = client["certificationDB_test_final"]
9  diamondcertificationDB = db["diamondcertificationDB"]
10 platinumcertificationDB = db["platinumcertificationDB"]
11 goldcertificationDB = db["goldcertificationDB"]
12
13 # Import all the records per certification level
14 json_file = open("release_diam.json")
15 data = json.load(json_file)
16
17 # Retrieving each collection
18 diamond_db=diamondcertificationDB.find()
19 plat_db=platinumcertificationDB.find()
20 gold_db=goldcertificationDB.find()
21

```

```

22 # Constructing Pandas' DataFrame
23 df_diamond = pd.DataFrame(list(diamond_db))
24 df_platinum = pd.DataFrame(list(plat_db))
25 df_gold = pd.DataFrame(list(gold_db))
26
27 #Deleting unwanted fields
28 query = {'$unset':{'themes':''}}
29 diamondcertificationDB.update_many({},query)
30
31 #Inserting new fields for each topic
32 topic_array= ["topic_1", "topic_2", "topic_3"]
33 j = 0
34 for index, row in df_diamond.iterrows():
35     for line in data:
36         filter = {'track_name':line['song_name']}
37         for i in topic_array:
38             new_value = { '$set':{'i':{'topic':data[j]['song']
39                 [0][i]['topic'],'words':data[j]['song'][0][i]['words']}}}
40             diamondcertificationDB.update_one(filter,new_value)
41         j+= 1

```

Source code 5.21: Modifying MongoDB's data to correct topic modelling

As can be observed hereinabove, before inserting new values into MongoDB, we wrote a condition to make sure that we are in the right record: namely, **the name of the actual track in MongoDB had to correspond to the name of the track in the JSON file** which, as a reminder, contained all the sequence of words that helped us define the correct topic labels per song.

But that was not all. Now that we had decent results for the covered topics for each considered song, we needed to modify the DataFrame that we had in our program: indeed, in order for our model to run correctly, we only wanted to keep the topic label and not the sequence of words defining it.

```

1 # Function to replace words in a topic cell
2 def replace_words(topic_cell):
3     if isinstance(topic_cell, dict):
4         if 'topic' in topic_cell:
5             topic = topic_cell['topic']
6             if topic in dictionnaire:
7                 new_topic = dictionnaire[topic]
8                 topic_cell['topic'] = new_topic
9     return topic_cell
10
11 # Word replacement in columns topic_1, topic_2 and topic_3
12 df_full_copy['topic_1'] = df_full_copy['topic_1'].apply(lambda x: replace_words(x))
13 df_full_copy['topic_2'] = df_full_copy['topic_2'].apply(lambda x: replace_words(x))
14 df_full_copy['topic_3'] = df_full_copy['topic_3'].apply(lambda x: replace_words(x))

```

Source code 5.22: Replacing values in *topic\_1*, *topic\_2* and *topic\_3*

After this replacement, our topics were processed in the *array\_dummiifier\_topics2()*'s

function that we presented before<sup>19</sup>, which transformed them into dummy variables.

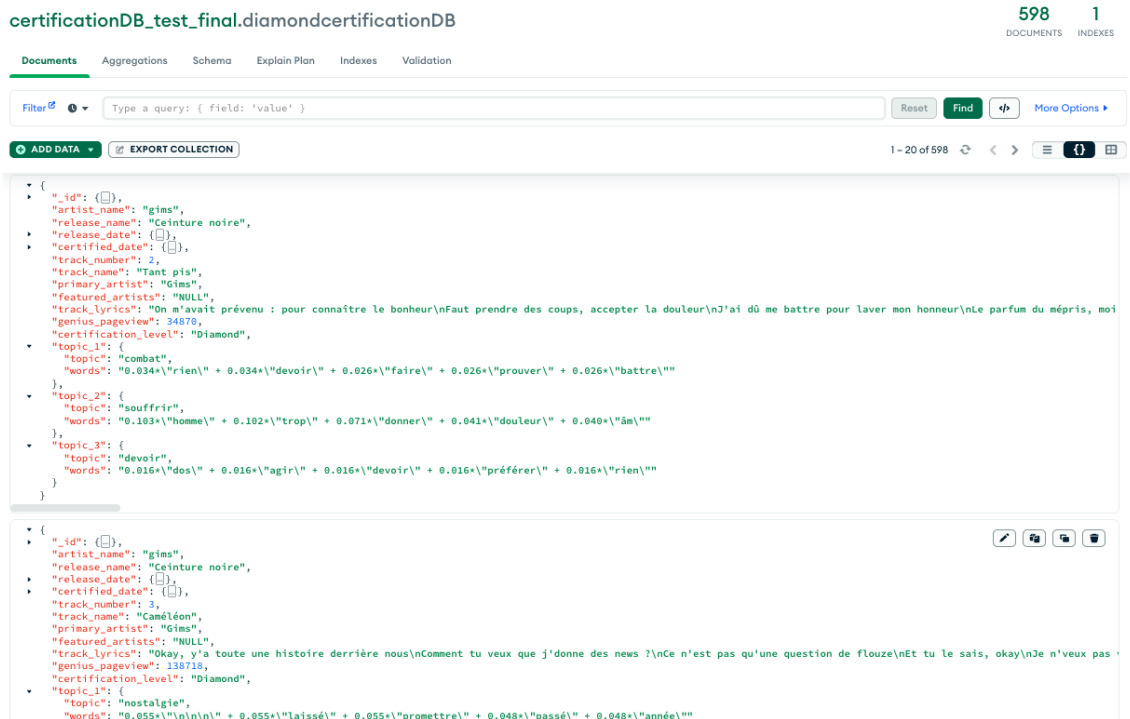


Figure 5.13: Updated topic labels in MongoDB

## Getting the training model right

One very simple yet crucial mistake we made was not to split the dataset we had into the features we want to make analysis from on one side, the target variable on the other. This was due to a lack of understanding that blocked us for a relative long period of time.

```
1 #Selecting our feature & target variables
2 feature_diam = df_full_copy.drop('certification_level_Diamond',axis=1)
3 target_diam = df_full_copy['certification_level_Diamond']
```

Source code 5.23: Our initial training model variable selection

Instead, as one can see above, we only dropped a specific dummy variable column (one of the three possible certifications), which obviously led to a very poor result, as the target variable should have rather been treated as a class variable.

<sup>19</sup>See 5.12

Certification level	Model evaluator	Test size	Accuracy	Precision	Recall	F1 score	K-C score	AUC	Train score	Test score
Diamond	KNN	0,900	0,970	0,950	0,650	0,770	0,760	0,987	0,980	0,970
Platinum	KNN	0,900	0,960	0,940	0,980	0,960	0,930	0,990	0,980	0,960
Gold	KNN	0,900	0,970	0,970	0,970	0,970	0,950	0,990	0,990	0,970
Diamond	KNN	0,750	0,980	0,940	0,780	0,850	0,840	0,990	0,986	0,980
Platinum	KNN	0,750	0,975	0,960	0,980	0,970	0,950	0,997	0,987	0,975
Gold	KNN	0,750	0,980	0,979	0,985	0,980	0,967	0,998	0,996	0,980
Diamond	KNN	0,500	0,980	0,970	0,800	0,870	0,870	0,980	0,990	0,980
Platinum	KNN	0,500	0,980	0,970	0,990	0,980	0,960	0,997	0,990	0,982
Gold	KNN	0,500	0,990	0,980	0,990	0,988	0,978	0,998	0,996	0,989
Diamond	Decision Tree	0,900	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Platinum	Decision Tree	0,900	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Gold	Decision Tree	0,900	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Diamond	Decision Tree	0,750	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Platinum	Decision Tree	0,750	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Gold	Decision Tree	0,750	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Diamond	Decision Tree	0,500	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Platinum	Decision Tree	0,500	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Gold	Decision Tree	0,500	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Diamond	Random Forest	0,900	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Platinum	Random Forest	0,900	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Gold	Random Forest	0,900	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Diamond	Random Forest	0,750	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Platinum	Random Forest	0,750	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Gold	Random Forest	0,750	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Diamond	Random Forest	0,500	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Platinum	Random Forest	0,500	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Gold	Naive Bayes	0,900	0,989	1,000	0,855	0,920	0,910	0,930	1,000	0,989
Platinum	Naive Bayes	0,900	0,990	1,000	0,986	0,990	0,987	1,000	0,996	0,993
Gold	Naive Bayes	0,900	0,980	1,000	0,960	0,980	0,960	1,000	0,989	0,980
Diamond	Naive Bayes	0,750	1,000	1,000	1,000	1,000	1,000	1,000	0,999	1,000
Platinum	Naive Bayes	0,750	0,993	1,000	0,985	0,990	0,980	1,000	0,994	0,993
Gold	Naive Bayes	0,750	0,990	1,000	0,980	0,990	0,980	0,999	0,990	0,990

Figure 5.14: A result sample based on our incorrect code

Figure 5.14 shows this poor result. We can observe that although several different usually very efficient training algorithms have been tested, none really gives a satisfactory, overfitting risk-free outcome.

A decent outcome is only reached once the code is fixed:

```

1 #Selecting our feature & target variables
2 feature = df_full_copy.drop('certification_level',axis=1)
3 target = df_full_copy['certification_level']

```

Source code 5.24: Our variable selection, fixed

Model evaluator	Test size	Accuracy	Precision	Recall	F1 score	K-C score	AUC	Train score	Test score
KNN	0,200	0,620	0,618	0,620	0,619	0,322		0,730	0,620
Decision Tree	0,200	0,594	0,593	0,594	0,593	0,283		0,970	0,594
Random Forest	0,200	0,656	0,658	0,656	0,652	0,376		0,970	0,656
Naive Bayes	0,200	0,324	0,661	0,324	0,366	0,144		0,311	0,323
SVM - linear kernel	0,200	0,646	0,602	0,646					
Logistic Regression - diam	0,200		0,540	0,120	0,190				
Logistic Regression - plat	0,200		0,620	0,720	0,670				
Logistic Regression - gold	0,200		0,680	0,660	0,670				
SVM - rbf kernel	0,200	0,651	0,660	0,651					
SVM - sigmoid kernel	0,200	0,507	0,530	0,507					
SVM - poly kernel	0,200	0,633	0,638	0,633					
Ordinal logistic Regression	0,200	0,649	0,648	0,649					

Figure 5.15: Result sample with variable selection for model training fixed

Figure 5.15 shows our second set of tests made on training algorithms. We can observe the main model performance measures: accuracy, precision, recall, F1-score, etc. Here, as discussed at p. 37, we made the choice to **prioritize the F1-score maximisation, as it**

**tends to encompass both precision and recall**, which makes it generally a pretty good key performance indicator.

Based on the F1-score, we can observe that indeed, **the Random Forest algorithm is the best algorithm among those which we tested to predict accurately on average our certification levels based on the features of the song.**

### Topic dummification

Another issue encountered regarding topic modelling originated from our need to dummify our class variables, which turned out to be problematic for topics and featurings<sup>20</sup>, as they both had too many empty columns to display in our model, which ineluctably induced overfitting.

More specifically, topics were duplicated, and sometimes even tripled, due to their particular form: indeed, *topic\_1*, *topic\_2* and *topic\_3*, which all three represented topics discussed in each song, were exclusive between them per song (meaning, a song could have no more than one occurrence of any specific topic), but non-exclusive between them from song to song (meaning, if "love" appeared in song 1 at position *topic\_1*, nothing prevented it from appearing in *topic\_2* in song 2). This, obviously, needed to be dealt with, since it did not bring any useful complementary information.

Our first strategy followed the next reasoning:

For each song, for

$$topic(i)$$

where

$$i \in \{1, 2, 3\}$$

1. Take *topic(i)*
2. Retrieve the topic inside of the cell
3. Fill the cell with only the topic label
4. Dummify the topic label
5. Repeat until end of database

To understand why we needed step 3, one must remember that our topic columns contained the topic label but also the sequence of words that caused the topic label.

This algorithm can be found in the Appendices, p. 81.

This strategy did not function, because Pandas' DataFrame caused an error. After many search, we concluded that the origin of this error was the lack of behavioral definition in our algorithm when we met the case where there were conflicting names. To solve that, we then proceeded to create suffices, so as to avoid conflicting column names (see Appendices, p. 81). The aim behind it was to proceed once again step by step: above all, we needed to keep all of our topic columns, since we did not want to lose information. Then, when we would dummify everything, we would be able to make a RegEx query for all column names having the label we desired, and join them together. Joint was possible following the next idea in an conflicting environment:

---

<sup>20</sup>For this issue, please refer to p. 60.



1. If topic was not yet encountered in our song: the value in this cell will simply go from "0" to "1"
2. If topic was already stated for this song: it is an error, since we designed our topic columns to be exclusive song per song

Given the above, we concluded that **the maximal value for a dummified topic column in our DataFrame could never be greater than 1, and joining same-label topic columns could only mean making an addition.**

The resulting algorithm, called *join\_labels()*, is found at Appendices, p. 82.

However, that still was not enough: as one can observe hereunder, the reached results were not as expected.

<u>What we expect</u>					<u>What we have</u>				
	love	love	war	war		love	love	war	war
0	1	1	0	0	0	1	1	0	1
1	1	1	1	1	1	1	1	0	1

Figure 5.16: Result from a small example code written on an online Python compiler (Programiz)

We spent a lot of time trying to figure out how to correctly sum on the X axis, but it somehow failed. So we had to take another approach.

Here below is the code that fixed this issue.

```

1 def array_dummifier_topics2(data):
2     # Unique topics extraction
3     topics = set()
4     for col in ['topic_1', 'topic_2', 'topic_3']:
5         topics.update(data[col].apply(lambda x: x['topic']
6             if isinstance(x, dict) else '').unique())
7
8     # Removal of null or None values from topics
9     topics.discard(None)
10    topics.discard('null')
11
12    # Creation of binary columns for each topic
13    for topic in topics:
14        data['topic_' + topic] = data.apply(lambda row:
15            1 if any(row[col]['topic'] == topic for col in
16                ['topic_1', 'topic_2', 'topic_3'] if
17                    isinstance(row[col], dict)) else 0, axis=1)
18
19    # Removal of topic_1, topic_2 and topic_3
20    df = data.drop(['topic_1', 'topic_2', 'topic_3'], axis=1)
21
22    return df

```

Source code 5.25: Fixing the duplicated topic columns

What this code makes differently, is that it uses Python's *lambda* to define two variables, namely *x* and *row*, which are used iteratively to respectively extract the topic from each concerned cell and to encode binary values in the right columns after one last verification.

And this approach works.

For precedent versions and trials to reach this code, please refer to the Appendices.

### Bigrams and extreme values filtering

At the very end of July, we detected a considerable mistake in our running code. As one can see here below, during the LDA topic modelling phase (see p. 31, we try to pre-process text content by filtering each word, considered as a "token", to respect two conditions: being 2-letters long at least and not being a stop word. But, what we should do instead is **consider sentences instead of separated words**, so as to allow bigram formations, correct dictionary definition and eventually, better topic modelling.

```
docs = []
for token in spacy_docs:
    tokens = []
    if len(token.orth_) > 2 and not token.is_stop:
        tokens.append( token.lemma_.lower() )
    docs.append(tokens)
```

Figure 5.17: Misuse of *tokens* list

This is a mistake that occurred at model creation, because we had trouble understanding what the correct algorithm tried to do; we actually thought this part was irrelevant, because we thought its use was to split documents when there were several to be given as input.

But in our model, *tokens* serve no use, since it reinitialized at each word passing, never taking more than a word at once.

```
1 docs = []
2     ##Splitting spacy_docs into sentences
3     for sent in spacy_docs.sents:
4         tokens = []
5         for token in sent:
6             if len(token.orth_) > 2 and not token.is_stop:
7                 tokens.append( token.lemma_.lower() )
8             ##Sending to docs not words, but list of words (= sentences)
9             docs.append(tokens)
10
11     ##min_count reduced: words must appear at least 2 times in docs
12     bigram = Phrases(docs, min_count=2)
13     for index in range(len(docs)):
14         for token in bigram[docs[index]]:
15             if '_' in token:
16                 docs[index].append(token)
```

Source code 5.26: Correct bit of code for pre-processing text contents

As seen just above, we also used this occasion to correct the parameters given to the bigrams creator function, since our initial *min\_count* = 10 parameter, which filtered words and bigrams which appeared less than 10 times in our text contents, was way too high of a threshold for our values.

We then tested the results of our new thematic modeling on a small sample of Diamond tracks to verify their accuracy. Obviously, this work was highly subjective, which the task of defining the words that best represent each song inherently tends to suggest. The result can be seen on the next page.

As expected, the labels we are able to retrieve from this version of the algorithm are better. We finally observe some bigrams, too, as well as diversified vocabulary. Regarding our filtering parameters, here is the set we have chosen:

- For *Phrases()*: *min\_count* = 2
- For *Dictionary.filter\_extremes()*: *no\_below* = 2, *no\_above* = 0.5

For other parameters combination results, please see the Appendices.

However, implementing this change would have meant re-labelling all of our sequence of words for all three levels of certification. The first time, this took us nearly 3 weeks of extensive workload. Since this error was observed very late in the research thesis, and in regards with the time it would take to correct it, this unfortunately was not implemented.

Still, as we will discuss in section 5.3, one can observe already very satisfying prediction capabilities in our model, even without the bigrams contribution. However, this will be clearly specified in our limitations.

## Quantity of topics analyzed

Another issue related to the topic modelling we had was the quantity of topics that we analyzed. As stated previously, these labels were put manually and at first, no concern was given to the amount of different labels possible. The first objective was to label all records and decrease amount of different labels later, if needed.

This need was met once all records were labeled. At that moment, we had no less than 332 different topics, which was too much to be relevant.

To see the total number of unique topics we had, we re-started from our full DataFrame (meaning: the frame with Diamond, Platinum and Gold releases), created a list of topics named `topics`, looped through our whole DataFrame and added those never encountered before.

```
topics = ['abandon', 'acteur', 'addiction', 'adulte', 'aide', 'aimer', 'alcool', 'alien', 'amitié', 'amour', 'améliorer', 'amériqu
e', 'animal', 'apparence', 'appeller', 'apprendre', 'argent', 'armée', 'arnaque', 'art', 'asiatique', 'astrologie', 'attendre', 'att
raction', 'authenticité', 'aventure', 'avion', 'bar', 'basketball', 'beau', 'besoin', 'bien-être', 'bizarre', 'boisson', 'bombe', 'b
onheur', 'business', 'caché', 'cadeau', 'calme', 'chance', 'changer', 'chaud', 'chercher', 'chimie', 'chômage', 'cinéma', 'cirque',
'club', 'cobaye', 'colère', 'combat', 'complexe', 'compétition', 'comédie', 'confession', 'confiance', 'connaître', 'conseiller', 'c
onsonner', 'contact', 'contrôle', 'corps', 'couleur', 'couple', 'courage', 'course', 'crier', 'criminalité', 'critique', 'croire',
'crédibilité', 'cuisine', 'culpabilité', 'célébrité', 'danger', 'danser', 'destinée', 'devoir', 'dialecte', 'difficulté', 'dire', 'd
iriger', 'dispute', 'distance', 'donner', 'dormir', 'drogue', 'durer', 'délicatesse', 'déranger', 'désir', 'détester', 'effort', 'eg
o', 'ennemi', 'ennui', 'envie', 'equipe', 'esclavage', 'espoir', 'espérer', 'etats-unis', 'excréments', 'explorer', 'exploser', 'ext
raordinaire', 'faiblesse', 'famille', 'fast_food', 'fatigue', 'feeling', 'femme', 'fin', 'fini', 'finir', 'flirt', 'folie', 'footbal
l', 'france', 'froid', 'fruit', 'fuir', 'fumer', 'futur', 'féminisme', 'fêter', 'gens', 'grandir', 'groupe', 'guerre', 'guérir', 'gé
rer', 'haine', 'handicap', 'histoire', 'homme', 'homophobie', 'honneur', 'horeca', 'identité', 'imaginer', 'inaction', 'incertitud
e', 'industrie', 'innocence', 'intelligent', 'internet', 'jalousie', 'jeu', 'jeu_hasard', 'jeunesse', 'lassitude', 'leader', 'libert
é', 'livre', 'luxe', 'magie', 'maison', 'mal', 'maladie', 'malheur', 'malin', 'maman', 'manga', 'manger', 'manifestation', 'manipula
tion', 'manque', 'marcher', 'mariage', 'maternité', 'mathématique', 'mentir', 'mer', 'merci', 'migration', 'monde', 'monstre', 'mora
le', 'mort', 'musculature', 'musique', 'médecin', 'médicament', 'méditation', 'méfiance', 'météo', 'nager', 'nature', 'nazisme', 'ne
ws', 'nostalgie', 'nuit', 'objectif', 'observer', 'ombre', 'ordinateur', 'origines', 'oublier', 'paix', 'papa', 'parcours', 'pardo
n', 'parler', 'partir', 'passion', 'passé', 'patience', 'pauvreté', 'pays', 'perdre', 'perdu', 'performance', 'perte', 'peur', 'phot
o', 'phrase', 'pirate', 'plaisir', 'pleurer', 'police', 'politique', 'popularité', 'pornographie', 'porte', 'pouvoir', 'prendre', 'p
rincipe', 'prison', 'problème', 'profiter', 'promener', 'promettre', 'propreté', 'protéger', 'pédophilie', 'racisme', 'radio', 'ra
p', 'rap_us', 'rareté', 'rater', 'refus', 'regretter', 'religion', 'renaître', 'repos', 'respect', 'ressentir', 'retour', 'retrait
e', 'retrouvailles', 'revenir', 'richesse', 'rire', 'robot', 'rouler', 'route', 'routine', 'royauté', 'rumeur', 'rupture', 'réfléchi
r', 'réseaux', 'réussir', 'réussite', 'réveil', 'révolution', 'rêver', 's'installer', 'santé', 'santé_mentale', 'sauver', 'savoir',
'secte', 'sens', 'signal', 'silence', 'skate', 'société', 'solitude', 'sortir', 'souffrir', 'souhaiter', 'souvenir', 'sport', 'stalk
er', 'stress', 'style', 'succès', 'super_héros', 'surprise', 'système', 'sécurité', 'sérénité', 'talent', 'temps', 'tendance', 'terr
orisme', 'trahison', 'traumatisme', 'travailler', 'tristesse', 'téléphone', 'télévision', 'tête', 'unité', 'victoire', 'vie', 'vieil
lesse', 'ville', 'violence', 'virilité', 'vivre', 'voiture', 'vol', 'volonté', 'voyager', 'véhicule', 'vérité', 'vêtements', 'week-
end', 'zone', 'échec', 'école', 'écouter', 'écrire', 'équipe', 'érotisme', 'été', 'événement']
Number of labels : 332
```

Tested song	Current model words output	Current model results	New model words output	New model results	Best topic modeler
GIMS - Tant pis	1. rien, devoir, faire, prouver, battre 2. homme, trop, donner, douleur, âme 3. dos, agir, pouvoir, préférer, rien	1. danger.violence 2. colère_mort_souffrir 3. objectif	1. insurmontable, rien, rien_insurmontable, pis, paix 2. pis, insurmontable, rien_insurmontable, rien, disent 3. pis, faire_faute, dos_poids, lot	1. colère_mort_souffrir 2. bien-être_santé_sport	New model
Nekfeu - N*que les clones (pt.2)	1. clone, nique, bon, prendre, sortir 2. ici, penser, nan, indique, confort 3. assiette, conscience, violent, bel, embrasser	1. religion_morale 2. ennui_routine_zone 3. danger.violence	1. n*que, n*que_clone, clone, code, sentir 2. dire, bon, jamais, voir, prendre 3. voir, monde, école, clone, donner	1. religion_morale 2. passé	Current model
Nekfeu - Tricheur ft. Damso	1. baba, mal, ville, trop, deviner 2. demain, succès, bang, bendo, guerre 3. falloir, annai, braise, évacuer, tokyo	1. famille_maman 2. richesse_succès 3. profiter_érotisme_voyager	1. trop, falloir, vie, sale, fesse 2. bientôt, changer, arrêter, ouais, mamma 3. bientôt, changer, bientôt_arrêter, arrêter, baba	1. famille_maman 2. changer_espérer_futur_rêver 3. profiter_érotisme_voyager	New model
Nekfeu - 1er rôle	1. vie, jamais, aimer, script, film 2. acteur, écrire, homme, crochetée, terre 3. vie, générique, rôle, donner, tête	1. art_culture_sport	1. tête, oeil, vis, cut, coupe 2. aimer, code, figu, devenir, réalisateur 3. cut, acteur, jamais, vie, mektoub_écrire	1. art_culture_sport 2. amour_femme_flirt	New model
Booba - DKR	1. quartier, quitte, vouloir, lion, tailler 2. maillé, bordeaux, casser, monsieur, être 3. mula, quitte, maillé, vouloir, n*gro	1. ennui_routine_zone 2. travailler 3. profiter_érotisme_voyager	1. mailler, lion, tailler, élan, monsieur 2. marier, vouloir, casser, dos, plat 3. mailler, quitte, quartier, dégât, mailler	1. travailler 2. amour_femme_flirt 3. profiter_érotisme_voyager	Equal

Table 5.2: Random songs comparison for both topic modelling algorithms

Figure 5.18: All topics we had after topic labeling

This led to a reduction of topics in several different manners:

- **Re-grouping under one word labels that were quite similar**

Example: *jeunesse* and *passé* were considered linked enough to go under one label; although both could have served as the "umbrella" topic, *passé* was considered more suitable as it restrained less the field of possible interpretations, which we were looking for here.

To do so, we simply replaced wrote *passé* as the replacement value of *jeunesse*.

- **Erasing non-relevant labels**

Example: *magie* was one of those words which was very distinguishable, potentially providing us more information than words such as *amour* or *drogues*, which we found quite often; however, its use was so anecdotal that erasing it did not seem as a loss of information at all.

To erase a label, we simply put "null" as its replacement value.

- **Re-grouping several labels together with an underscore ( \_ ) when concepts were pretty close one to another**

Example: *amour*, *femme* and *flirt* were all strongly linked, yet they all proposed a slight nuance that was considered interesting to keep, giving us therefore: *amour\_affection\_flirt*.

This was done by hand, in our replacement list file.

```
≡ all_mindmap_wip_test.txt
1  abandon : colère_mort_souffrir
2  acteur : art_culture_sport
3  addiction : colère_mort_souffrir
4  adulte : changer_espérer_futur_rêver
5  aide : colère_mort_souffrir
6  aimer : amour_femme_flirt
7  alcool : profiter_érotisme_voyager
8  alien : null
9  amitié : amitié_équipe
10 amour : amour_femme_flirt
11 améliorer : null
12 Amérique : origines_pays
13 animal : null
14 apparence : null
15 appeler : null
16 apprendre : null
17 argent : business_industrie
18 armée : police_prison_système
19 arnaque : danger_violence
20 art : art_culture_sport
21 asiatique : origines_pays
22 astrologie : null
23 attendre : null
24 attraction : amour_femme_flirt
25 authenticité : religion_morale
```

Figure 5.19: Preview of our topics replacement list

```

1  ''' Creating a dico and replacing themes '''
2
3  #Loading text file containing our dictionary
4  with open('all_mindmap_wip_test.txt', 'r') as file:
5      lines = file.readlines()
6
7  #Creating the dictionary based on text file lines
8  dictionnaire = {}
9  for line in lines:
10     line = line.strip().split(' ')
11     mot = line[0]
12     valeur = line[2] if len(line) > 1 else None
13     dictionnaire[mot] = valeur
14
15  #Pop empty first value
16  if ' ' in dictionnaire:
17     dictionnaire.pop(' ')
18
19  #Replace word function
20  def replace_words(topic_cell):
21     if isinstance(topic_cell, dict):
22         if 'topic' in topic_cell:
23             topic = topic_cell['topic']
24             if topic in dictionnaire:
25                 new_topic = dictionnaire[topic]
26                 topic_cell['topic'] = new_topic
27     return topic_cell
28
29  #Replacing words in topic_1, topic_2, topic_3 columns
30  df_full_copy['topic_1'] = df_full_copy['topic_1'].
31     apply(lambda x: replace_words(x))
32  df_full_copy['topic_2'] = df_full_copy['topic_2'].
33     apply(lambda x: replace_words(x))
34  df_full_copy['topic_3'] = df_full_copy['topic_3'].
35     apply(lambda x: replace_words(x))

```

Source code 5.27: Full code for themes grouping and filtering

This fortunately worked, as we passed from 332 topics in total to as low as 20, which was a great sign for our prediction model.

### Too many columns for *featuring*

One of the reasons that led to having an overfit was the quantity of columns in our DataFrame.

track_id	track_name	album_name	album_release_date	album_genre	album_artists	album_artists_list	album_artists_list_str	album_artists_list_str_lower	album_artists_list_str_lower_lower	album_artists_list_str_lower_lower_lower	album_artists_list_str_lower_lower_lower_lower	album_artists_list_str_lower_lower_lower_lower_lower	album_artists_list_str_lower_lower_lower_lower_lower_lower	album_artists_list_str_lower_lower_lower_lower_lower_lower_lower	album_artists_list_str_lower_lower_lower_lower_lower_lower_lower_lower	album_artists_list_str_lower_lower_lower_lower_lower_lower_lower_lower_lower	album_artists_list_str_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower	album_artists_list_str_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower	album_artists_list_str_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower	album_artists_list_str_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower	album_artists_list_str_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower	album_artists_list_str_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower_lower	
1	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000	1.0000000000000000

Figure 5.20: Our gigantic database with features dummified

And, after some reflection, we decided to change our *featuring* variable, as we observed the tremendous amount of dummy variables produced by the latter. Indeed, initially, this variable was considering each guest artist on a track as a separate feature, treating it therefore as a dummy variable, which led to an absurd amount of columns with 0-filled values. The model was so big and buggy that we had to store it in a CSV file before loading it, as re-running our program would take too much time.

We ultimately reached the conclusion that it was a big problem, as much on the efficiency side as well as on the practical side, since it could definitely be a major reason for our model overfitting. Thus, this needed to be changed.

But this still could be an interesting explanatory variable; so, instead of simply erasing it, the decision we took was **to make it binary**:

- *featuring* = 1 if we had artist(s) featured on the track
- *featuring* = 0 otherwise

This way, we hoped to observe a potential impact (or not) of other artist's presence on the success of tracks.

And here is the code to do this conversion:

```

1 ''' STEP 5.2/ FEATURINGS AS BINARY VARIABLE '''
2
3 df_full_copy["featured_artists"] = df_full_copy["featured_artists"].
4                                     replace('NULL',0)
5
6 # Function to replace values different from 0 to 1
7 def replace_non_zero(value):
8     if value != 0:
9         return 1
10    else:
11        return value
  
```

```

12
13 # Application of the function to the right column
14 df_full_copy['featured_artists'] = df_full_copy['featured_artists'].
15                                     apply(replace_non_zero)

```

Source code 5.28: Making featurings a binary variable

## 5.3 Results

Our model displayed very interesting outcomes, as it pointed **relatively strong prediction capabilities**.

But first of all, let us proceed by analyzing different parameters such as variable independence and correlation between features.

Table 5.3, on next page, displays the statistical tests done on some of our features. As one can observe, there are in total 4 statistical tests realized on different combinations of features to help us highlight variable dependence, homoscedasticity, etc. To sum it up, the outcome shows us **a small dependence between featuring presence and record certification and a negative correlation between the release day and the number of views on a song**.

Table 5.4, on next page too, in turn, represents all the Chi-square tests done on the combination of the record certification and the topics we have determined. We can once again observe **variable dependence** among the following sets of features:

- Certification level & topic\_bien-être\_santé
- Certification level & topic\_changer\_espérer\_futur\_rêver
- Certification level & topic\_passé
- Certification level & topic\_origines\_pays
- Certification level & topic\_homophobie\_racisme
- Certification level & topic\_objectif

To interpret it, the results inform us the said topics show a greater probability to have a release of a specific certification, which is indeed interesting for our prediction ability.

Regarding the date split into year, month and day, a possible hypothesis would be that the scientific research observations were proven correct, in the sense that **the release date of a music project could indeed influence the short-, mid- or long-term success of an album**, and therefore of its constituting songs. The common example in that regard is given by albums releasing just before the Christmas holidays, which, as one knows, is a very significant commercial period of time, where sales tend to meaningfully increase. Another example would be the release time in the month: potentially, some albums could benefit from an early-month release, as people tend to consume more in that time, which also impacts leisure and cultural goods consumption. Indubitably, our first model, which only had the release year, did not capture that.



Statistical test	Chi-square		Student			ANOVA		Pearson					
Null hypothesis (H0)	Variable independence		Population means are equal			Population means are equal		Absence of variable correlation					
Tested features	Featured artists & certification level	Topics & certification level	Pageviews & certification levels (Diamond and Platinum)	Pageviews & certification levels (Platinum and Gold)	Pageviews & certification levels (Diamond and Gold)	Pageviews & certification levels (Diamond, Platinum and Gold)	Year & Month	Year & Day	Year & pageviews	Month & Day	Month & pageviews	Day & pageviews	
Value	10,30812	See next table.	19,13255	14,06746	28,36269	372,38536	0,05342	-0,14144	0,06466	-0,155216	0,04739	-0,00286	
p-value	0,00577	See next table.	3,46188e-77	3,76574e-44	4,12698e-157	1,07649e-152	5,59726e-05	8,62323e-27	1,06477e-06	5,51327e-32	0,00035	0,82939	
Reject H0?	Yes	See next table.	No	No	No	No	No	No	No	No	No	Yes	

Table 5.3: Results of the statistical tests

Statistical test	Chi-square																				
Null hypothesis (H0)	Variable independence																				
Tested topic	internet, technologies	bien-être, santé	profiter, érotisme, voyager	amitié, équipe	religion, morale	danger, violence	changer, espérer, futur, rêver	ennui, routine, zone	business, industrie	art, culture, sport	police, prison, système	colère, mort, souffrir	passé	rap	travailler	origines, pays	amour, femme, flirt	homophobie, racisme	famille, maman	richesse, succès	objectif
Value	19,39873	5,25221	139,83756	23,50454	15,19912	77,23542	2,34087	24,17891	23,52855	13,16805	31,88252	59,88099	9,79801	30,83447	1,36879	2,49546	51,26056	0,94329	41,48562	49,50958	10,15869
p-value	6,13225e-05	0,07236	4,31179e-31	7,87143e-06	0,00050	1,69255e-17	0,31023	5,16844e-06	7,77750e-06	0,001382	1,19343e-07	9,93132e-14	0,007454	2,01549e-07	0,50439	0,28715	7,39452e-12	0,62397	9,80645e-10	1,77472e-11	0,00622
Reject H0?	No	Yes	No	No	No	No	Yes	No	No	No	No	No	Yes	No	Yes	Yes	No	Yes	No	No	Yes

Table 5.4: Results of Chi-square realized on *topic\_x* and *certification\_level* variables

### 5.3.1 Making predictions

To further test our prediction ability, we decided to use the `predict()` function of our current Random Forest Tree model. This simple function enabled us to very easily visualize the power of our model.

```

1  '''STEP 12/ MAKING PREDICTIONS'''
2  df_full_copy['certification_level_prediction'] = rf.predict(feature)
3  df_full_copy = df_full_copy.reindex(sorted(df_full_copy.columns), axis=1)
4
5  # Save new dataframe into csv file
6  df_full_copy.to_csv('predictions.csv', index=False)

```

Source code 5.29: Predicting the level of certification a record will get

certification_level	certification_level_prediction	day	featured_artists	genius_pageview	month	topic_	topic_amitié_équipe	topicamour_femme_flirt	topic_art_culture_sport	topic_bien-être_santé
Diamond	Diamond	0.7703916092402157	0	-0.2921440423106577	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.030054322654638494	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	1.0516360519911878	-1.0106437866808586	0.0	0.0	1.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.17394222454352848	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	1	1.0172421145337451	-1.0106437866808586	0.0	0.0	0.0	1.0	0.0
Diamond	Diamond	0.7703916092402157	1	3.2440464510992837	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.058750657510863474	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.19426402102428889	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	1	2.1789905761868114	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.12726771402057233	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.2584271508961567	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	1	0.3409252280674337	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.1805043628228104	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.16765090466942326	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	1	-0.24786523306426483	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.24932348601521637	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.23569923701632625	-1.0106437866808586	0.0	0.0	1.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	0.3134163277569835	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.04373065211606259	-1.0106437866808586	0.0	0.0	1.0	0.0	1.0
Diamond	Diamond	0.7703916092402157	0	-0.28151962795372504	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	0.012307925570603793	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	0.08689756401167514	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.245448699602688	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.18186887094120077	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	1	-0.20837782637099844	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.12914261067179575	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.12686148641280726	-1.0106437866808586	0.0	0.0	1.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	0.5091972025147348	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	0	-0.24645906057584727	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	1	0.42450395434197064	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0
Diamond	Diamond	0.7703916092402157	1	-0.2180647924023194	-1.0106437866808586	0.0	0.0	0.0	0.0	0.0

Figure 5.21: Sample of our `.csv` prediction file

The full `.csv` file can be found at [GitHub, 2023].

This, as expected, corroborates totally results we have reached statistically. The model is therefore usable.

### 5.3.2 Storing our model

For further re-usability, we follow the guideline in [Atha, 2021] and we store our model into a `.lib` file. What this file allows is re-using this model on any computer and by-passing the training step.

```

1  ''' STEP 13/ SAVING THE MODEL'''
2  from joblib import dump, load
3
4  # Saving model
5  dump(rf, 'certification_level_classification.joblib')

```

```
6 # Loading model
7 clf = load('certification_level_classification.joblib')
```

Source code 5.30: Saving our model in a *.joblib* file

## 5.4 Discussion

The result of our model tend to confirm several hypotheses:

1. **A song commercial success can be at least partially predicted by its features:** we observe very strong prediction capabilities in our model when we take into account the release time, the presence of featurings or not, the covered topics and the online popularity of the said song.
2. **The release year time plays a role:** as proposed by [Bhattacharjee et al., 2007], the release date indeed seems to play a significant role in the success of an album. We observe it as we progressed from initially only taking the release year, to then year and month, to finally consider the year, the month, but also the day. All the three together summed proved to be more accurate predictors than just by taking one of them separately.
3. **Analysing singles certification could have been a strong plus:** among many other features to try, we think that taking singles certification into account could have been a very meaningful improvement. Indeed, when we think of song success, the certification comes to mind very quickly. This was planned but unfortunately, we were not able to achieve that in time.

Now that we have achieved a fully working prediction model, we consider it important to talk about limitations regarding our results and future research paths.

## Chapter 6

# Conclusions

## 6.1 Limitations

This thesis tried to propose a prediction model for music success. However, hypotheses were naturally taken to accomplish this objective. In this section, we are going to highlight these hypotheses to describe their limitations.

1. **We focused on French national charts to predict Hip-Hop success:** here, we can already highlight several restrictions taken during this thesis: the geographical, language and music genre constraints in first place. We needed to put a frame to our research by limiting our scope. We believe that our observations, extended to other genres or countries, could lend meaningful results.
2. **We retrieved albums, but we analyzed songs:** unfortunately, due to a lack of time, this thesis neither used singles present in the corresponding albums to analyse their impact on the latter, potentially leading to a commercial album success analysis, nor did it encapsulated singles certifications as a variable in our model. Both could have potentially improved or nuanced our predictions, and we surely think they are very interesting hypothesis to try.
3. **Prediction accuracy could have been boosted by the album effect:** what we mean here by "the album effect" is that the songs of the same album reach the same success level, as we have concentrated on the success of albums and not singles. This has not been tested, but it could potentially be a wrong reason for which prediction is that good.
4. **Our topic modelling could have been improved:** as discussed in section 5.25, we could have taken into account information given by bigrams in our topic modelling algorithm. Unfortunately, we made a mistake that we failed to correct on time, but we managed to still retrieve satisfying results.

## 6.2 Future research paths

The nature and scope of this thesis inevitably put aside some potentially relevant features and explanatory criterias. Here, we propose to highlight the most important ones we observed, based on scientific literature.

1. **Importance of the artist's popularity:** a very interesting feature would be to take into account the popularity of the artist we are analyzing. Scientific literature has shown us this approach makes sense, and it would be particularly fitting to use the quantity of publicly available data nowadays to improve the predictions. Of course, the first step would be to correctly define what we intend by "popularity". As shown in [Lee et al., 2003], one definition could be the total number of sales the artist made throughout his or her whole career, as well as the number of certifications his or her received. Another interesting approach would be to take into account its social network presence and popularity, as this domain is playing a major role in the discovery and renown of artists today.
2. **Importance of the featured artist(s):** now that we know the featuring has some impact in the commercial success of a release, it would be interesting to carry on this path by looking at the specific popularity of the featuring. This could also be combined to a point stated above, which is correlate it with the popularity of the featured artist (for instance its total number of sales, or its social network popularity, etc).

3. **Importance of the musical characteristics of the song:** as we have seen in [Kellaris and Kent, 2023], assessing the different components of a song result in useful information. Therefore, re-examining the tonality, the texture, the tempo and potentially more characteristics of a song in the light of available data on Spotify for instance, such as its genre or its total duration, its spectre and so on, could be beneficial for our model.
4. **Importance of the music label:** the music industry is divided in roughly two categories: independent, small labels and massive major labels. As seen in [Bhattacharjee et al., 2007], the methods and means of one category are not equal to the other one; it would therefore be interesting to test this difference more in depth.
5. **Importance of the duration before certification:** the quantity of available projects today is bigger than ever, with the rise of streaming audio services and the digitalisation of music production. In that regard, one interesting question could be: are albums certified faster than before<sup>1</sup>? Does that play a role in music's commercial success? Were past artists less prone to certification? Are older releases taking advantage of the duration effect nowadays by winning more certifications?
6. **Importance of the album:** in this thesis, we made the choice to analyze each song separately, although they all belong to certified albums. But what if we analyzed certified songs instead? Some reach certification even if the albums they were put out never did, or they simply do not belong to any album. Moreover, an interesting analysis could be to reflect on the impact of the album for the constituting songs: meaning, if we have two very successful songs in the same album, for instance, could that potentially increase the chances for this album to be successful, too? What effects can create a short or long tracklist?
7. **Importance of certification categorization:** at last, we have focused on the main three levels of certification in our study, starting from the rarest (and thus, the most successful): diamond, platinum and gold. But what about the non-certified albums? They could have been added to our set, yet they were left out. Certifications have also sub-categorization; some albums can be 1x platinum, or 2x platinum, just to cite this simple example. Could that impact our predictions, too? It is certainly worth testing.

As one can understand, this field is very fruitful and could lead other very important discoveries.

---

<sup>1</sup>Table 7.14 makes a small-set comparison of best selling Hip-Hop albums in the United States across three decades: 2001, 2011 and 2021, and checks the time needed to reach Platinum certification. One very distinguishable observation is although artists tend to reach Platinum quite fast (going from several months to more or less one year) both in 2001 and 2021, the tendency is quite different in 2011: in average, a top-10 selling album needs 4.5 years to reach Platinum. One explanation path could be the economic crisis and the consequent downward trend in the music industry before streaming accountability in 2016. This all lead to consider this field very interesting to analyze.

## Chapter 7

# Appendices

## 7.1 Developments

### 7.1.1 Methodology

#### Data collection and analysis

```
1 #Loading SMOTE library
2 from imblearn.over_sampling import SMOTE
3 smote = SMOTE()
4
5 #Oversampling
6 X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
7
8 #Running RF with oversampling
9 rf = RandomForestClassifier()
10 rf.fit(X_train_resampled, y_train_resampled)
```

Source code 7.1: Declaration of the Random Forest algorithm with oversampling technique

```
1 #Loading SMOTE library
2 from imblearn.over_sampling import SMOTE
3 smote = SMOTE()
4
5 #Oversampling
6 X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
7
8 #Running DT with oversampling
9 dtc = tree.DecisionTreeClassifier()
10 dtc.fit(X_train_resampled, y_train_resampled)
```

Source code 7.2: Declaration of the Decision Tree algorithm with oversampling technique

```
1 #Loading SMOTE library
2 from imblearn.over_sampling import SMOTE
3 smote = SMOTE()
4
5 #Oversampling
6 X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
7
8 #Running NB with oversampling
9 nb = GaussianNB()
10 nb.fit(X_train_resampled, y_train_resampled)
```

Source code 7.3: Declaration of the Gaussian Naive Bayes algorithm with oversampling technique

```
1 #Loading SMOTE library
2 from imblearn.over_sampling import SMOTE
3 smote = SMOTE()
```



	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,17	0,88	0,28	148
<b>Platinum</b>	0,75	0,58	0,65	951
<b>Gold</b>	0,62	0,36	0,46	972
<hr/>				
<b>Accuracy</b>			0,5	2071
<b>Macro average</b>	0,51	0,61	0,47	2071
<b>Weighted average</b>	0,65	0,5	0,54	2071

Table 7.1: Classification report: hard Voting Classifier (LogReg, RF & SVC)

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,22	0,8	0,34	158
<b>Platinum</b>	0,72	0,6	0,65	925
<b>Gold</b>	0,63	0,47	0,54	988
<hr/>				
<b>Accuracy</b>			0,55	2071
<b>Macro average</b>	0,52	0,62	0,51	2071
<b>Weighted average</b>	0,64	0,55	0,57	2071

Table 7.2: Classification report: hard Voting Classifier (DT, RF & SVC)

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,18	0,84	0,3	145
<b>Platinum</b>	0,71	0,57	0,63	919
<b>Gold</b>	0,62	0,41	0,49	1007
<hr/>				
<b>Accuracy</b>			0,51	2071
<b>Macro average</b>	0,51	0,61	0,47	2071
<b>Weighted average</b>	0,63	0,51	0,54	2071

Table 7.3: Classification report: hard Voting Classifier (RF, SVC & KNN)

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,19	0,87	0,31	160
<b>Platinum</b>	0,71	0,61	0,65	936
<b>Gold</b>	0,63	0,34	0,44	975
<hr/>				
<b>Accuracy</b>			0,5	2071
<b>Macro average</b>	0,51	0,61	0,47	2071
<b>Weighted average</b>	0,63	0,5	0,53	2071

Table 7.4: Classification report: hard Voting Classifier (RF, SVC & NB)

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,17	0,86	0,28	128
<b>Platinum</b>	0,72	0,58	0,64	952
<b>Gold</b>	0,62	0,41	0,49	991
<b>Accuracy</b>				
			0,51	2071
<b>Macro average</b>	0,5	0,61	0,47	2071
<b>Weighted average</b>	0,64	0,51	0,55	2071

Table 7.5: Classification report: hard Voting Classifier (RF, DT, SVC, NB & KNN)

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,21	0,78	0,33	149
<b>Platinum</b>	0,76	0,57	0,65	981
<b>Gold</b>	0,59	0,49	0,53	941
<b>Accuracy</b>				
			0,55	2071
<b>Macro average</b>	0,52	0,61	0,5	2071
<b>Weighted average</b>	0,64	0,55	0,57	2071

Table 7.6: Classification report: soft Voting Classifier (RF,DT, SVC, NB & KNN)

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,17	0,9	0,29	156
<b>Platinum</b>	0,75	0,6	0,67	971
<b>Gold</b>	0,61	0,31	0,41	944
<hr/>				
<b>Accuracy</b>			0,49	2071
<b>Macro average</b>	0,51	0,6	0,46	2071
<b>Weighted average</b>	0,64	0,49	0,52	2071

Table 7.7: Classification report: soft Voting Classifier (RF, SVC & NB)

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,21	0,85	0,33	146
<b>Platinum</b>	0,73	0,59	0,66	940
<b>Gold</b>	0,64	0,45	0,53	985
<hr/>				
<b>Accuracy</b>			0,55	2071
<b>Macro average</b>	0,52	0,63	0,51	2071
<b>Weighted average</b>	0,65	0,55	0,57	2071

Table 7.8: Classification report: soft Voting Classifier (LR, RF & SVC)

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,23	0,75	0,35	134
<b>Platinum</b>	0,71	0,6	0,65	975
<b>Gold</b>	0,6	0,51	0,55	962
<b>Accuracy</b>				
			0,57	2071
<b>Macro average</b>	0,51	0,62	0,52	2071
<b>Weighted average</b>	0,63	0,57	0,58	2071

Table 7.9: Classification report: soft Voting Classifier (DT, RF &amp; SVC)

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,2	0,79	0,32	152
<b>Platinum</b>	0,72	0,58	0,65	954
<b>Gold</b>	0,6	0,44	0,51	965
<b>Accuracy</b>				
			0,53	2071
<b>Macro average</b>	0,51	0,6	0,49	2071
<b>Weighted average</b>	0,63	0,53	0,56	2071

Table 7.10: Classification report: soft Voting Classifier (RF, SVC &amp; KNN)

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,57	0,35	0,44	141
<b>Platinum</b>	0,68	0,68	0,68	943
<b>Gold</b>	0,65	0,68	0,67	987
<b>Accuracy</b>				
			0,66	2071
<b>Macro average</b>	0,63	0,57	0,59	2071
<b>Weighted average</b>	0,66	0,66	0,66	2071

Table 7.11: Classification report: oversampling RF

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,62	0,58	0,6	160
<b>Platinum</b>	0,78	0,75	0,76	930
<b>Gold</b>	0,73	0,76	0,74	981
<b>Accuracy</b>				
			0,74	2071
<b>Macro average</b>	0,71	0,7	0,7	2071
<b>Weighted average</b>	0,74	0,74	0,74	2071

Table 7.12: Classification report: oversampling RF + month/year variable split

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Diamond</b>	0,74	0,66	0,7	154
<b>Platinum</b>	0,88	0,82	0,85	941
<b>Gold</b>	0,8	0,87	0,83	976
<hr/>				
<b>Accuracy</b>			0,83	2071
<b>Macro average</b>	0,81	0,78	0,79	2071
<b>Weighted average</b>	0,83	0,83	0,83	2071

Table 7.13: Classification report: oversampling RF + day/month/year variable split

```

4
5 #Oversampling
6 X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
7
8 #Running KNN with oversampling
9 nn = KNeighborsClassifier()
10 knn.fit(X_train_resampled, y_train_resampled)

```

Source code 7.4: Declaration of the K-nearest Neighbors algorithm with oversampling technique

```

1 #Loading SMOTE library
2 from imblearn.over_sampling import SMOTE
3 smote = SMOTE()
4
5 #Oversampling
6 X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
7
8 #Running LR with oversampling
9 lr = LogisticRegression()
10 lr.fit(X_train_resampled, y_train_resampled)

```

Source code 7.5: Declaration of the Logistic Regression algorithm with oversampling technique

```

1 #Loading SMOTE library
2 from imblearn.over_sampling import SMOTE
3 smote = SMOTE()
4
5 #Oversampling
6 X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
7
8 #Running SVC with oversampling
9 clf = svm.SVC()
10 clf.fit(X_train_resampled, y_train_resampled)

```

Source code 7.6: Declaration of the Support Vector Classifier algorithm with oversampling technique

```

1 #Loading SMOTE library
2 from imblearn.over_sampling import SMOTE
3 smote = SMOTE()
4
5 #Oversampling
6 X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
7
8 #Running RF with oversampling
9 rf = RandomForestClassifier(random_state=0)
10 rf.fit(X_train_resampled, y_train_resampled)

```



Source code 7.7: SMOTE oversampling

```
1 #Loading RandomUnderSampler
2 from imblearn.under_sampling import RandomUnderSampler
3
4 #Undersampling
5 rus = RandomUnderSampler()
6 X_train_resampled, y_train_resampled = rus.fit_resample(X_train, y_train)
7
8 #Running RF with undersampling
9 rf = RandomForestClassifier(random_state=0)
10 rf.fit(X_train_resampled, y_train_resampled)
```

Source code 7.8: *imblearn*'s undersampling

```
1 #Loading SMOTEENN
2 from imblearn.combine import SMOTEENN
3 smote_enn = SMOTEENN()
4
5 #Resampling
6 X_train_resampled, y_train_resampled = smote_enn.fit_resample(X_train, y_train)
7
8 #Running RF with SMOTEENN
9 rf = RandomForestClassifier(random_state=0)
10 rf.fit(X_train_resampled, y_train_resampled)
```

Source code 7.9: SMOTEENN sampling

```
1 #Loading libraries
2 from sklearn.ensemble import RandomForestClassifier, VotingClassifier
3 from sklearn.linear_model import LogisticRegression
4 from sklearn.svm import SVC
5 from imblearn.combine import SMOTEENN
6
7 #Defining SMOTEENN
8 smote_enn = SMOTEENN()
9
10 #Resampling
11 X_resampled, y_resampled = smote_enn.fit_resample(X_train, y_train)
12
13 # Models definition
14 clf1 = RandomForestClassifier()
15 clf2 = SVC(probability=True)
16 clf3 = KNeighborsClassifier()
17
18 #Creating the final model
19 eclf = VotingClassifier(estimators=[('rf', clf1), ('svc', clf2), ('knn', clf3)],
20                             voting='soft')
21
```

22  
23  
24

```
#Training the final model
ec1f.fit(X_resampled, y_resampled)
```

Source code 7.10: One example of Voting Classifier model

Scenario analysis - MT model				
Aa Name	Definition	Advantages	Weaknesses	Status
Regression model	Predicting future happenings between a dependent and one or more independent variables based on their relationship on known datasets	<ul style="list-style-type: none"> <li>- Good to find relationship between variables (<a href="#">appier</a>)</li> <li>- Fast inference, interpretable coefficients (<a href="#">crunchingthedata</a>)</li> <li>- We can transform our qualitative data into quantitative (1 or 0 for each category)</li> </ul>	<ul style="list-style-type: none"> <li>- Estimates a numerical value (≠ classification models: estimates a category) (<a href="#">appier</a>)</li> <li>- Thrown off by outliers (<a href="#">crunchingthedata</a>)</li> <li>- Not very good at predicting (<a href="#">crunchingthedata</a>)</li> </ul>	👎 not for me
Clustering analysis	Unsupervised machine learning method identifying and grouping similar data points in larger datasets without concern for the specific outcome	<ul style="list-style-type: none"> <li>- Great at initially separating data into subsets (<a href="#">explorium</a>)</li> <li>- Hierarchical clustering tends to produce more accurate results (<a href="#">explorium</a>)</li> <li>- Contrary to classification, it groups objects based on similarities, not predefined classes (<a href="#">explorium</a>)</li> </ul>	<ul style="list-style-type: none"> <li>- Results may be more related to the arbitrary structure than to the data itself (<a href="#">explorium</a>)</li> <li>- Hierarchical clustering requires significant computation power, not ideal when working with larger datasets (<a href="#">explorium</a>)</li> <li>- Hierarchical clustering is sensitive to outliers: can produce inaccurate clusters therefore (<a href="#">explorium</a>)</li> </ul>	👍 can be used
Network analysis	Studying the relationships between entities in a network. Analysing the connections, or links, between the entities, as well as the characteristics of the entities themselves.	<ul style="list-style-type: none"> <li>- It can help identify key influencers (<a href="#">mygreatlearning</a>)</li> <li>- We can make predictive analysis with it (<a href="#">mygreatlearning</a>)</li> </ul>	<ul style="list-style-type: none"> <li>- The accent is strongly put on spread of information, flows, etc. : this is not my case</li> <li>- I think the "network" term must be taken literally: the social networks, etc., but not a database</li> </ul>	👎 not for me
Classification model	Used to assign items to a discrete group or class based on a specific set of features	<ul style="list-style-type: none"> <li>- We can supervise the model (<a href="#">simplilearn</a>)</li> <li>- We can define what is the category goal to reach: diamond certification, and therefore manage to work with qualitative data (<a href="#">simplilearn</a>)</li> </ul>	<ul style="list-style-type: none"> <li>- Surely, I work with qualitative data, but I also do have quantitative one: could it be a problem?</li> </ul>	👍 that's it

Figure 7.1: Sample of our statistical model analysis

### Scientific literature review

```
1 #Features importance computation
2 feature_importances = rf.feature_importances_
3
4 #Features importance displaying
5 for name, importance in zip(X_train.columns, feature_importances):
6     print(f"Feature: {name}, Importance: {importance}")
```

Source code 7.11: Feature importance analysis code

## Problems encountered and solutions found

```
1 def array_dummiifier_topics(dataframe, collection):
2     # Opening JSON
3     f = open("json_dumps/release_"+ collection + ".json")
4     data = json.load(f)
5
6     # Looping - retrieving labels
7     topic_array = ['topic_1', 'topic_2', 'topic_3']
8     for row in data:
9         for index, line in dataframe.iterrows():
10            if (row['song_name'] == line['track_name']):
11                for topic in topic_array:
12                    temp = row['song'][0][topic]['topic']
13                    # if not temp == "NaN":
14                    if temp != "NaN":
15                        dataframe.loc[dataframe[topic] == line[topic], topic] = temp
16
17            df1 = (dataframe[col["topic"].explode().str.get_dummies().groupby(level=0).sum().
18                add_prefix('topic_'))
19            dataframe = dataframe.drop(col, axis=1).join(df1)
20
21            f.close()
22            return dataframe
```

Source code 7.12: Our *array\_dummiifier\_topics()* function

```
1 # Looping - changing fields in the DF
2     for topic in topic_array:
3         df1 = (dataframe[topic].explode().str.get_dummies().groupby(level=0).
4             sum().add_prefix('topic_')) #creating a new column
5
6         if topic == 'topic_3':
7             dataframe = dataframe.drop(topic, axis=1).join(df1, rsuffix='_right2',
8                 how='outer')
9         else:
10            dataframe = dataframe.drop(topic, axis=1).join(df1, lsuffix='_left',
11                rsuffix='_right', how='outer')
```

Source code 7.13: Addition to *array\_dummiifier\_topics()* to deal with conflicting column names

```
1 def join_labels(df):
2     labels = []
3
4     #For topics, we change their name from `topic_name` to simply `name`
5     for col in df.columns:
6         if (re.match('^topic_.*', col)):
7             test = re.search("(?<=_.)+(?=_)|(?<=_)?!left)?!right).*", col)
8             label = 'topic_'+test.group()
9             df.rename(columns = {col:label}, inplace = True)
```

```

10     if not label in labels:
11         labels.append(label)
12
13     #Adding columns, erasing duplicates
14     for label in labels:
15         #We sum our similar columns
16         df[label]= df.loc[:, label].sum(axis=1)
17
18     #We dump duplicates
19     df = df.loc[:,~df.T.duplicated(keep='first')]
20     return df

```

Source code 7.14: Our function `join_labels()`, working in pair with `array_dummifier_topics()`

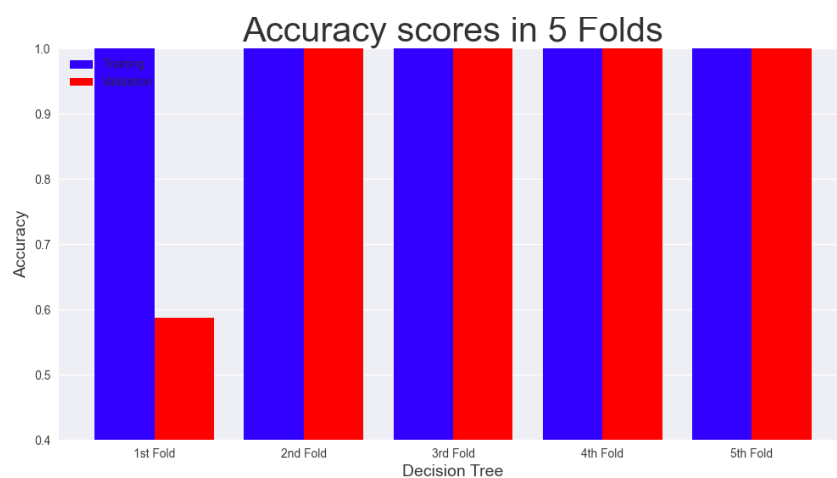


Figure 7.2: Our initial k-fold cross-validation result with Decision Tree (overfit)

```

1  {'Training Accuracy scores': array([1., 1., 1., 1., 1.]),
2  'Mean Training Accuracy': 100.0,
3  'Training Precision scores': array([1., 1., 1., 1., 1.]),
4  'Mean Training Precision': 1.0,
5  'Training Recall scores': array([1., 1., 1., 1., 1.]),
6  'Mean Training Recall': 1.0,
7  'Training F1 scores': array([1., 1., 1., 1., 1.]),
8  'Mean Training F1 Score': 1.0,
9  'Validation Accuracy scores': array([0.58682266, 1., 1., 1., 1.]),
10 'Mean Validation Accuracy': 91.73645320197045,
11 'Validation Precision scores': array([1., 1., 1., 1., 1.]),
12 'Mean Validation Precision': 1.0,
13 'Validation Recall scores': array([0.09690444, 1., 1., 1., 1.]),
14 'Mean Validation Recall': 0.8193808882907133,
15 'Validation F1 scores': array([0.17668712, 1., 1., 1., 1.]),
16 'Mean Validation F1 Score': 0.8353374233128834}

```

Source code 7.15: Output of the Decision Tree k-fold cross-validation method

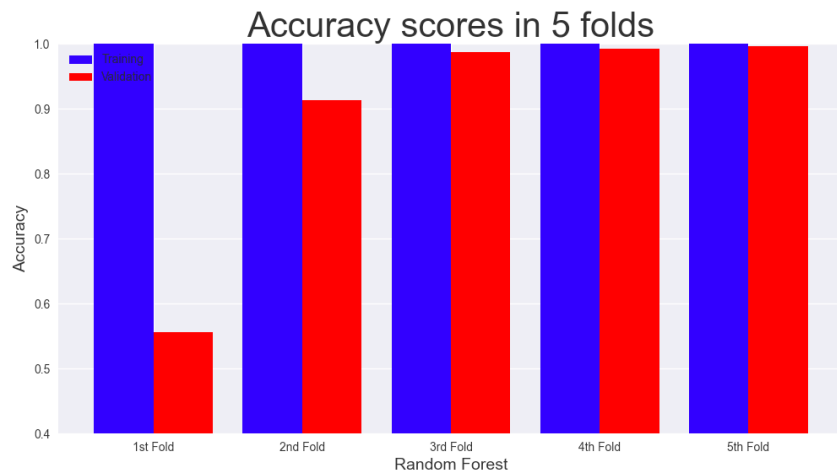


Figure 7.3: Our k-fold cross-validation result with Random Forest (overfit)

```

1 {'Training Accuracy scores': array([1., 1., 1., 1., 1.]),
2   'Mean Training Accuracy': 100.0,
3   'Training Precision scores': array([1., 1., 1., 1., 1.]),
4   'Mean Training Precision': 1.0,
5   'Training Recall scores': array([1., 1., 1., 1., 1.]),
6   'Mean Training Recall': 1.0,
7   'Training F1 scores': array([1., 1., 1., 1., 1.]),
8   'Mean Training F1 Score': 1.0,
9   'Validation Accuracy scores': array([0.55603448, 0.91256158, 0.98706897, 0.99261084,
10    0.99630542]),
11  'Mean Validation Accuracy': 88.89162561576353,
12  'Validation Precision scores': array([1.          , 0.84108967, 0.99048913, 0.98539177,
13    0.99198932]),
14  'Mean Validation Precision': 0.961791977324754,
15  'Validation Recall scores': array([0.02960969, 0.99730821, 0.98115747, 0.9986541 ,
16    1.]),
17  'Mean Validation Recall': 0.8013458950201884,
18  'Validation F1 scores': array([0.05751634, 0.91256158, 0.98580122, 0.99197861,
19    0.99597855]),
20  'Mean Validation F1 Score': 0.7887672590333978}

```

Source code 7.16: Initial output of the Random Forest k-fold cross-validation method

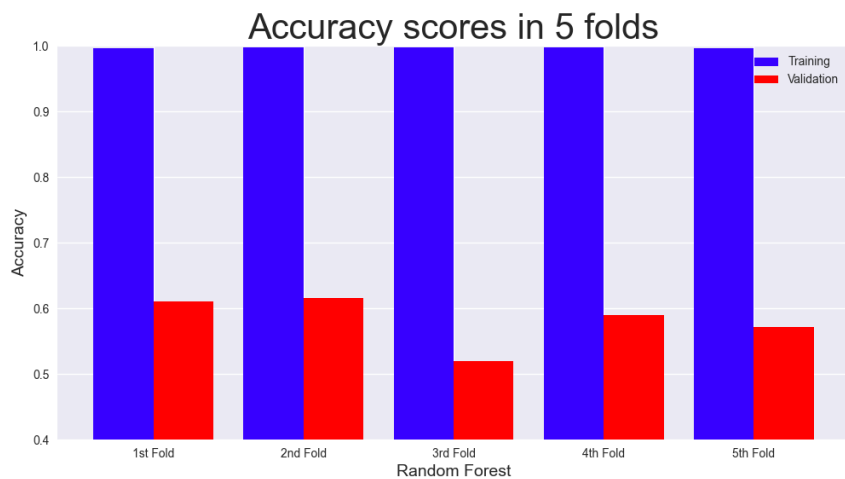


Figure 7.4: Our final result of k-fold cross-validation with Random Forest and the over-sampling technique applied

```
1 {'Training Accuracy scores': array([0.9966133 , 0.99784483, 0.99676724, 0.99769089,  
2     0.99645936]),  
3 'Mean Training Accuracy': 99.70751231527093,  
4 'Training Precision scores': array([nan, nan, nan, nan, nan]),  
5 'Mean Training Precision': nan,  
6 'Training Recall scores': array([nan, nan, nan, nan, nan]),  
7 'Mean Training Recall': nan,  
8 'Training F1 scores': array([nan, nan, nan, nan, nan]),  
9 'Mean Training F1 Score': nan,  
10 'Validation Accuracy scores': array([0.61083744, 0.61576355, 0.51970443, 0.58990148,  
11     0.57142857]),  
12 'Mean Validation Accuracy': 58.1527093596059,  
13 'Validation Precision scores': array([nan, nan, nan, nan, nan]),  
14 'Mean Validation Precision': nan,  
15 'Validation Recall scores': array([nan, nan, nan, nan, nan]),  
16 'Mean Validation Recall': nan,  
17 'Validation F1 scores': array([nan, nan, nan, nan, nan]),  
18 'Mean Validation F1 Score': nan}
```

Source code 7.17: Our final output for the oversampled Random Forest k-fold cross-validation method

## 7.2 Conclusions

### 7.2.1 Future research paths

Year	2001	Release date	Certified after	2011	Release date	Certified after	2021	Release date	Certified after
#1	Ludacris - Word of Mouf	27/11/01	Platinum after 2 months	Drake - Take care	15/11/11	Platinum after 2 months	Drake - Certified Lover Boy	3/09/21	Platinum after 1 month
#2	Ja Rule - Pain is love	2/10/01	Platinum after 1 month	Jay-Z & Kanye West - Watch the throne	12/08/11	5x plat after 9 years, 3 months	Pop Smoke - Shoot for the Stars Aim for the Moon	2/07/20	Platinum after 1 year
#3	Jay-Z - The Blueprint	11/09/01	Platinum after 1 month	Beyoncé - 4	28/07/11	Platinum after 1 month	Doja Cat - Planet Her	24/06/21	Platinum after 1 year
#4	2Pac - Until the end of time	27/03/01	Platinum after 2 months	Lil Wayne - Tha Carter IV	28/07/11	Platinum after 7 years, 4 months	Lil Baby - My Turn	28/02/20	Platinum after 3 months
#5	D12 - Devil's night	19/06/01	Platinum after 3 months	Chris Brown - F.A.M.E.	18/03/11	Platinum after 5 years	Rod Wave - Soulfly	26/03/21	Platinum after 1 year, 4 months
#6	Missy Elliott - Miss E... so addictive	26/04/01	Platinum after 3 months	Wiz Khalifa - Rolling papers	29/03/11	Platinum after 5 years, 3 months	Moneybagg Yo - A Gangsta's Pain	23/04/21	Platinum after 7 months
#7	DMX - The great Depression	23/10/01	Platinum after 2 months	J. Cole - Cole World	20/09/11	Platinum after 4 years, 5 months	Juice Wrld - Legends Never Die	10/07/20	Platinum after 1 year, 3 months
#8	Nas - Illmatic	19/04/94	Platinum after 7 years, 7 months	Big Sean - Finally Famous	28/06/11	Platinum after 6 years, 4 months	Polo G - Hall of Fame	11/06/21	Platinum after 7 months
#9	Busta Rhymes - Genesis	27/11/01	Platinum after 4 months	Young Jeezy - TM: 103 Hustlerz Ambition	20/12/11	Platinum after 8 years, 7 months	Post Malone - Hollywood's Bleeding	06/09/19	Platinum after 1 year, 7 months
#10	Fat Joe - J.O.S.E.	26/11/01	Platinum after 6 months	Bad Meets Evil : Hell: The Sequel	14/07/11	Gold after 2 months	Lil Durk - The Voice	24/12/20	Platinum after 1 year, 7 months

Table 7.14: 2001, 2011 and 2021 best-selling rap albums comparison

## 7.3 Additional appendices

```
1 from sklearn.model_selection import cross_validate
2 def cross_validation(model, _X, _y, _cv=5):
3     '''Function to perform 5 Folds Cross-Validation
4     Parameters
5     -----
6     model: Python Class, default=None
7         This is the machine learning algorithm to be used for training.
8     _X: array
9         This is the matrix of features.
10    _y: array
11        This is the target variable.
12    _cv: int, default=5
13        Determines the number of folds for cross-validation.
14    Returns
15    -----
16    The function returns a dictionary containing the metrics 'accuracy', 'precision',
17    'recall', 'f1' for both training set and validation set.
18    '''
19    _scoring = ['accuracy', 'precision', 'recall', 'f1']
20    results = cross_validate(estimator=model,
21                            X=_X,
22                            y=_y,
23                            cv=_cv,
24                            scoring=_scoring,
25                            return_train_score=True)
26
27    return {"Training Accuracy scores": results['train_accuracy'],
28            "Mean Training Accuracy": results['train_accuracy'].mean()*100,
29            "Training Precision scores": results['train_precision'],
30            "Mean Training Precision": results['train_precision'].mean(),
31            "Training Recall scores": results['train_recall'],
32            "Mean Training Recall": results['train_recall'].mean(),
33            "Training F1 scores": results['train_f1'],
34            "Mean Training F1 Score": results['train_f1'].mean(),
35            "Validation Accuracy scores": results['test_accuracy'],
36            "Mean Validation Accuracy": results['test_accuracy'].mean()*100,
37            "Validation Precision scores": results['test_precision'],
38            "Mean Validation Precision": results['test_precision'].mean(),
39            "Validation Recall scores": results['test_recall'],
40            "Mean Validation Recall": results['test_recall'].mean(),
41            "Validation F1 scores": results['test_f1'],
42            "Mean Validation F1 Score": results['test_f1'].mean()
43    }
44
45 # Grouped Bar Chart for both training and validation data
46 def plot_result(x_label, y_label, plot_title, train_data, val_data):
47     '''Function to plot a grouped bar chart showing the training and validation
48     results of the ML model in each fold after applying K-fold cross-validation.
49     Parameters
50     -----
51     x_label: str,
52         Name of the algorithm used for training e.g 'Decision Tree'
```



53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84

```

y_label: str,
    Name of metric being visualized e.g 'Accuracy'
plot_title: str,
    This is the title of the plot e.g 'Accuracy Plot'

train_result: list, array
    This is the list containing either training precision, accuracy, or f1 score.

val_result: list, array
    This is the list containing either validation precision, accuracy,
    or f1 score.

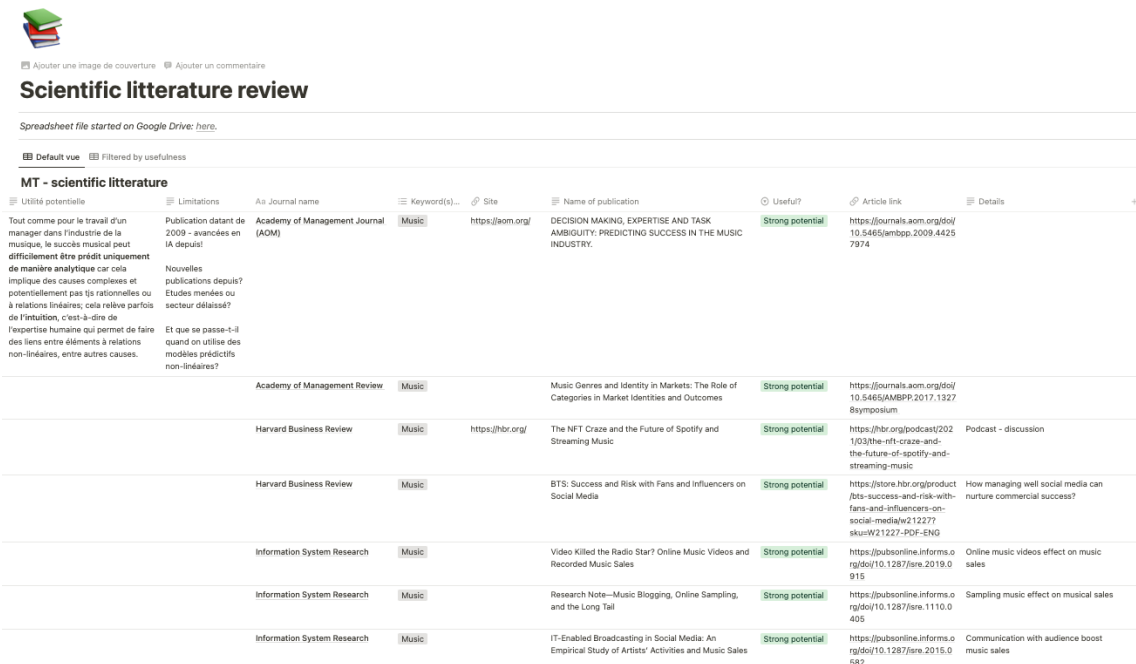
Returns
-----

The function returns a Grouped Barchart showing the training and validation
result in each fold.
'''

# Set size of plot
plt.figure(figsize=(12,6))
labels = ["1st Fold", "2nd Fold", "3rd Fold", "4th Fold", "5th Fold"]
X_axis = np.arange(len(labels))
ax = plt.gca()
plt.ylim(0.40000, 1)
plt.bar(X_axis-0.2, train_data, 0.4, color='blue', label='Training')
plt.bar(X_axis+0.2, val_data, 0.4, color='red', label='Validation')
plt.title(plot_title, fontsize=30)
plt.xticks(X_axis, labels)
plt.xlabel(x_label, fontsize=14)
plt.ylabel(y_label, fontsize=14)
plt.legend()
plt.grid(True)
plt.show()

```

Source code 7.18: Our k-fold cross-validation algorithm taken from [Affiah, 2022]



**Scientific literature review**

Spreadsheet file started on Google Drive: [here](#).

Default view | Filtered by usefulness

**MT - scientific literature**

Utilité potentielle	Limitations	Journal name	Keyword(s)	Site	Name of publication	Useful?	Article link	Details
Tout comme pour le travail d'un manager dans l'industrie de la musique, le succès musical peut difficilement être prédit uniquement de manière analytique car cela implique des causes complexes et potentiellement pas très rationnelles ou à relations linéaires, cela relève parfois de l'intuition, c'est-à-dire de l'expertise humaine qui permet de faire des liens entre éléments à relations non-linéaires, entre autres causes.	Publication datant de 2009 - avancées en IA depuis!	Academy of Management Journal (AMJ)	Music	https://aom.org/	DECISION MAKING, EXPERTISE AND TASK AMBIGUITY: PREDICTING SUCCESS IN THE MUSIC INDUSTRY.	Strong potential	https://journals.aom.org/doi/10.5465/ambpp.2009.44257974	
	Nouvelles publications depuis? Etudes menées ou secteur délaissé?	Academy of Management Review	Music		Music Genres and Identity in Markets: The Role of Categories in Market Identities and Outcomes	Strong potential	https://journals.aom.org/doi/10.5465/ambpp.2017.13278symposium	
	Et que se passe-t-il quand on utilise des modèles prédictifs non-linéaires?	Harvard Business Review	Music	https://hbr.org/	The NFT Craze and the Future of Spotify and Streaming Music	Strong potential	https://hbr.org/podcast/2021/03/the-nft-craze-and-the-future-of-spotify-and-streaming-music	Podcast - discussion
		Harvard Business Review	Music		BTS: Success and Risk with Fans and Influencers on Social Media	Strong potential	https://store.hbr.org/product/78e-success-and-risk-with-fans-and-influencers-on-social-media/w212277sku=W21227-PDF-ENG	How managing well social media can nurture commercial success?
		Information System Research	Music		Video Killed the Radio Star? Online Music Videos and Recorded Music Sales	Strong potential	https://pubsonline.informs.org/doi/10.1287/isre.2019.0915	Online music videos effect on music sales
		Information System Research	Music		Research Note—Music Blogging, Online Sampling, and the Long Tail	Strong potential	https://pubsonline.informs.org/doi/10.1287/isre.1110.0405	Sampling music effect on musical sales
		Information System Research	Music		IT-Enabled Broadcasting in Social Media: An Empirical Study of Artists' Activities and Music Sales	Strong potential	https://pubsonline.informs.org/doi/10.1287/isre.2015.0582	Communication with audience boost music sales

Figure 7.5: Sample of our scientific literature database on Notion

WORK ON SOURCES (outdated - see Notion)						
KEYWORD: MUSIC						
Name	Additional keyword	Site	Potentially useful publications	Useful?	Link	Details
Academy of Management Journal (ACM)		<a href="https://ajm.amsnet.org/">https://ajm.amsnet.org/</a>	<b>PERSONAL MAKING, EXPERTISE AND TASK AMBIGUITY: PREDICTING SUCCESS IN THE MUSIC INDUSTRY: Working to the Beat: A Self-Regulatory Framework Linking Music Characteristics to Job Performance</b>	Strong potential	<a href="https://journals.ama.com.org/doi/10.5465/ambpp.2009.44257974">https://journals.ama.com.org/doi/10.5465/ambpp.2009.44257974</a>	
Academy of Management Review	Success	<a href="https://om.oxfordjournals.org/">https://om.oxfordjournals.org/</a>	<b>How musical festival success is affected by differences in consumer and producer classification</b> <b>Music Genres and Identity in Markets: The Role of Categories in Market Identities and Outcomes</b> <b>How do Nested Categories Influence Market Emergence? Evidence from Early American Music Records</b> <b>Open Ties and Deep Success</b> <b>Spotify v. Pandora: Competing with Revenue Models and Activity Systems</b> <b>Conceptualizing and Measuring Outcomes in Creative Industries Research</b>	Maybe Strong potential Some potential Some potential Maybe Some potential	<a href="https://journals.ama.com.org/doi/10.5465/ambpp.2017.13279syposium">https://journals.ama.com.org/doi/10.5465/ambpp.2017.13279syposium</a> <a href="https://journals.ama.com.org/doi/10.5465/ambpp.2015.78">https://journals.ama.com.org/doi/10.5465/ambpp.2015.78</a> <a href="https://journals.ama.com.org/doi/10.5465/ambpp.2020.103">https://journals.ama.com.org/doi/10.5465/ambpp.2020.103</a> <a href="https://journals.ama.com.org/doi/10.5465/ambpp.2019.15462abstract">https://journals.ama.com.org/doi/10.5465/ambpp.2019.15462abstract</a> <a href="https://journals.ama.com.org/doi/10.5465/ambpp.2019.18949syposium">https://journals.ama.com.org/doi/10.5465/ambpp.2019.18949syposium</a>	
Accounting, Organizations and Society	Success	<a href="https://www.sciencedirect.com/journal/accounting-organizations-and-society">https://www.sciencedirect.com/journal/accounting-organizations-and-society</a>	<b>Combining creativity and control: Understanding individual motivation in large-scale collaborative creativity</b>	Maybe	<a href="https://www.sciencedirect.com/science/article/abs/pii/S0361368219300253">https://www.sciencedirect.com/science/article/abs/pii/S0361368219300253</a>	
Contemporary accounting research	Success	<a href="https://onlinelibrary.wiley.com/doi/10.1111/1368778x.12603">https://onlinelibrary.wiley.com/doi/10.1111/1368778x.12603</a>	<b>Making artworks valuable: Categorisation and modes of valuation work</b> <b>Is Music an Evolutionary Adaptation?</b>	Maybe Some potential	<a href="https://onlinelibrary.wiley.com/doi/10.1111/1368778x.12603">https://onlinelibrary.wiley.com/doi/10.1111/1368778x.12603</a> <a href="https://onlinelibrary.wiley.com/doi/10.1111/1368778x.12603">https://onlinelibrary.wiley.com/doi/10.1111/1368778x.12603</a>	
Harvard Business Review		<a href="https://hbr.org/">https://hbr.org/</a>	<b>The social context of musical success: A developmental account</b> <b>Music and emotions: from enchantment to entrainment</b> <b>Mental imagery in music performance: underlying mechanisms and potential benefits</b> <b>The NFT Craze and the Future of Spotify and Streaming Music</b> <b>The Music Industry in the 2020s</b> <b>The Birth of Tencent Music Entertainment</b> <b>Tencent Music Entertainment Group: Molding Music with Social Experiences</b> <b>TROODS INVESTMENT MANAGEMENT - YAMAHA: MAKING MUSIC SUSTAINABLE</b> <b>Apple and the Music Industry</b> <b>BTS: Success and Risk with Fans and Influencers on Social Media</b> <b>Live Nation and Pharrell Williams</b> <b>Spotify: Face the Music (Update 2019)</b>	Some potential Maybe Some potential Strong potential Some potential Some potential Some potential Maybe Some potential Strong potential Some potential Strong potential	<a href="https://onlinelibrary.wiley.com/doi/10.1111/1468-4632.12603">https://onlinelibrary.wiley.com/doi/10.1111/1468-4632.12603</a> <a href="https://onlinelibrary.wiley.com/doi/10.1111/1468-4632.12603">https://onlinelibrary.wiley.com/doi/10.1111/1468-4632.12603</a> <a href="https://onlinelibrary.wiley.com/doi/10.1111/1468-4632.12603">https://onlinelibrary.wiley.com/doi/10.1111/1468-4632.12603</a> <a href="https://hbr.org/podcast/2021/03/the-nft-craze-and-the-future-of-spotify-and-streaming-music">https://hbr.org/podcast/2021/03/the-nft-craze-and-the-future-of-spotify-and-streaming-music</a> <a href="https://hbr.org/product/the-music-industry-in-the-2020s/issue/68">https://hbr.org/product/the-music-industry-in-the-2020s/issue/68</a> <a href="https://hbr.org/product/the-birth-of-tencent-music-entertainment/issue/53888">https://hbr.org/product/the-birth-of-tencent-music-entertainment/issue/53888</a> <a href="https://hbr.org/product/tencent-music-entertainment-group-reel/issue/53888">https://hbr.org/product/tencent-music-entertainment-group-reel/issue/53888</a> <a href="https://hbr.org/product/troods-investment-management-yamaha-making-music-sustainable/issue/117734">https://hbr.org/product/troods-investment-management-yamaha-making-music-sustainable/issue/117734</a> <a href="https://hbr.org/product/apple-and-the-music-industry/issue/520374">https://hbr.org/product/apple-and-the-music-industry/issue/520374</a> <a href="https://hbr.org/product/bts-success-and-risk-with-fans-and-influencers-on-social-media/issue/122734">https://hbr.org/product/bts-success-and-risk-with-fans-and-influencers-on-social-media/issue/122734</a> <a href="https://hbr.org/product/live-nation-and-pharrell-williams/issue/521005">https://hbr.org/product/live-nation-and-pharrell-williams/issue/521005</a> <a href="https://hbr.org/product/spotify-face-the-music-update-2019/issue/73874">https://hbr.org/product/spotify-face-the-music-update-2019/issue/73874</a>	How education in childhood can determine artist Neural impact of music linked to our primary art Podcast: discussion China leading music company The follow-up How Live Nation can continue to create value for How managing well social media can nurture it
Information System Research			<b>Video Killed the Radio Star? Online Music Videos and Recorded Music Sales</b> <b>Research Note—Music Blogging, Online Sampling, and the Long Tail</b> <b>IT-Enabled Broadband in Social Media: An Emotional Study of Artists' Activities and Music Sales</b>	Strong potential Strong potential Strong potential	<a href="https://onlinelibrary.wiley.com/doi/10.1287/isre.2013.30915">https://onlinelibrary.wiley.com/doi/10.1287/isre.2013.30915</a> <a href="https://onlinelibrary.wiley.com/doi/10.1287/isre.2013.30915">https://onlinelibrary.wiley.com/doi/10.1287/isre.2013.30915</a> <a href="https://onlinelibrary.wiley.com/doi/10.1287/isre.2013.30915">https://onlinelibrary.wiley.com/doi/10.1287/isre.2013.30915</a>	Online music videos effect on music sales Sampling music effect on musical sales Communication with audience boost music sale

Figure 7.6: Sample of our scientific literature database on Google Spreadsheet

Ajouter une image de couverture Ajouter un commentaire

## Mémoire

- UNI LIFE
- Mémoire
- Stage

---

- Quand bosser dessus?
- Idées à développer
- Entretiens avec mon promoteur

---

- Development plan
- Gens intervenants
- My research onion

---

- Intermediary report (reflective assessment)
- Areas of concern
- Recap: table of contents

---

- To-do linked to the thesis
- Global agenda view
- Agenda
- Drafts
  - Organisation
  - Success: definition and limitations
    - "Success": definition and limitations
    - Diamond certified rap releases: a first code test
    - Building the database: Platinum and Gold releases
    - Topic modelling
  - Regression model analysis
    - Statistical model: definition of the type
    - Building the statistical model
    - Overfitting & cross-validation
  - Details for LaTeX file
    - Time before certification - 2021 vs 2011 vs 2001 (for US Hip-Hop)

Figure 7.7: Sample of our main page for the research thesis on Notion

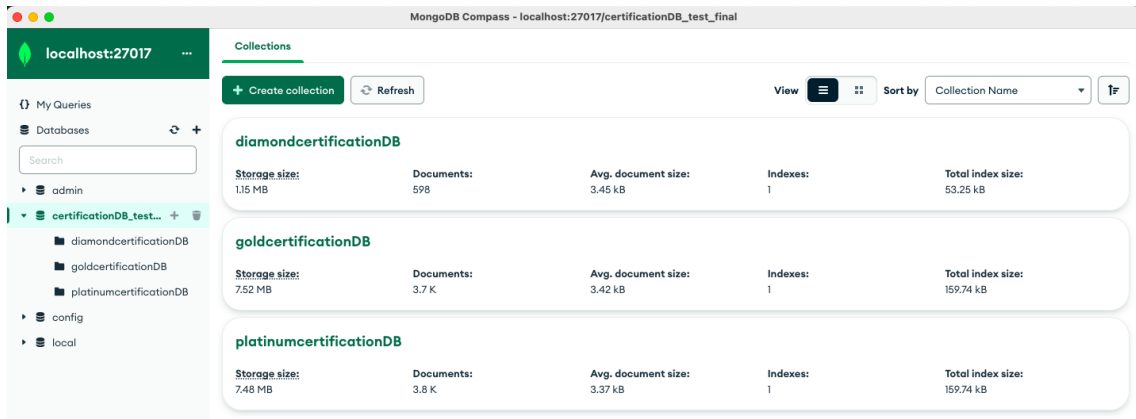


Figure 7.8: Overview of our MongoDB database

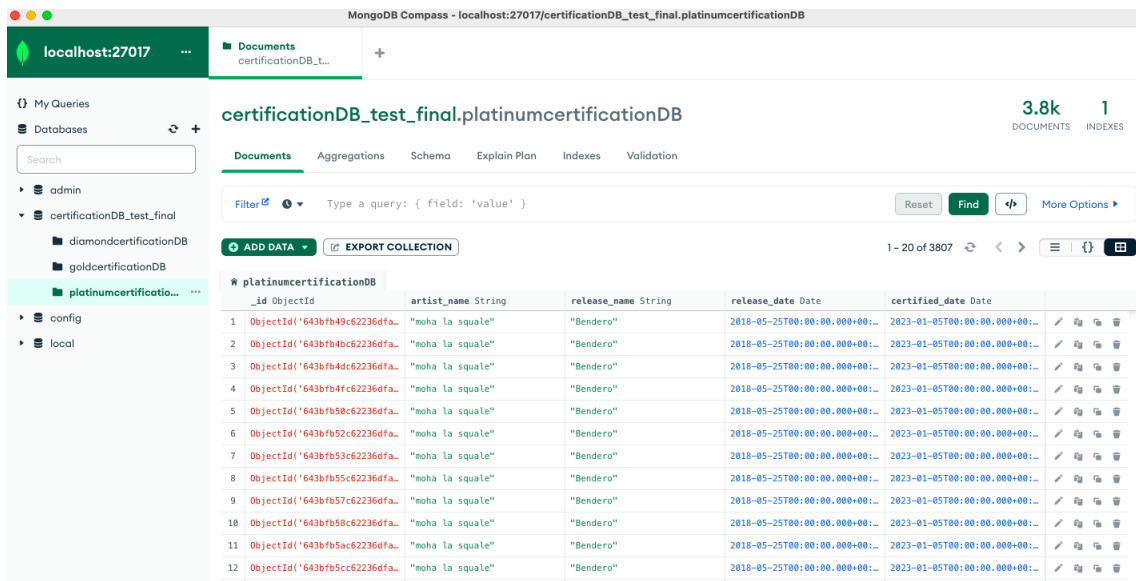


Figure 7.9: An overview of our Platinum MongoDB collection

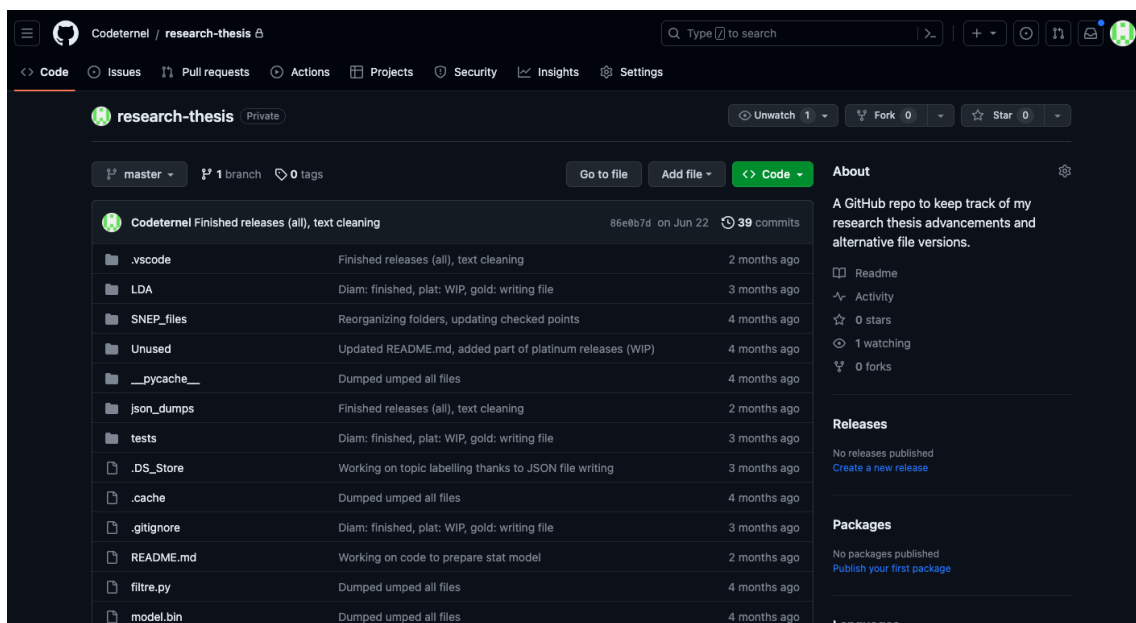


Figure 7.10: Overview of our GitHub page for the research thesis (available at [GitHub, 2023])

- ▼ Records to fix
  - Platinum records
    - Naps – "A l'instinct" is not found on Genius; rather, the artist "Agrab Al Nass" is found. How to fix it? (z=28, so I have to write z≤27)
      - Test: titre de projet remplacé par "Instinct"
    - Sadek – "#VVRDL" is not found on Genius since its full name is "Vulgaire, violent et ravi d'être là" (z=78)
      - Remplacé par "Vulgaire, violent et ravi d'être là"
    - \$-Crew – "Destins liés" is not found on Genius; Genius writes "S-Crew" instead (z=106)
      - Remplacé par "S-Crew"
    - Niro – "Les autres / Or game" is not found on Genius; it is the compilation of two albums, actually (z=153)
      - Splitté en deux records: "Les autres" et "Or game"
      - Puisque "Or game" est fourni en addition à "Les autres", on met les deux albums à la date de ce dernier
    - JUL – "Album gratuit" is mistaken on Genius; it takes the 2015 release, whereas it should take the "Album gratuit, vol.1" one
      - Remplacé par "Album gratuit, vol 1"
      - **Ne pas oublier d'effacer les mauvaises entrées dans la BDD!**
- + :: Gold records
  - DJ Cut Killer & DJ Abdel – "Cut Killer Show 2 & A l'ancienne"
    - Splitté en deux
  - Green Montana – "NOSTALGIA +" was not found in the JSON files, since the "+" is not given by Genius when download the JSON file.
    - Remplacé par "NOSTALGIA"
    - C'est dangereux de faire ça, mais ici je sais que Montana n'a pas d'autre album portant ce nom!
  - Soprano - "Coffret 4 CDs 2019" doesn't work, as it takes together:
    - Puisqu'il faut vivre
    - La colombe et le corbeau
    - Cosmopolitanie
    - L'Everest
    - **I must mention it somewhere in my work: the marketing campaigns around the albums (but here, we don't consider it)**
  - Luv Resval – "Etoile noire" was badly encoded; it stored "Etoile noire: Nébuleuse" instead of "Etoile noire: Brise-Monde"
    - Delete the JSON file
    - Delete the recording

Figure 7.11: By-hand correction to some releases

## Chapter 8

### List of resource persons

Below, one can find all the resource persons contacted during the realisation of this thesis, with the full name, mail address and field of expertise.

<b>First name</b>	<b>Last name</b>	<b>Mail address</b>	<b>Field of expertise</b>
Aswhin	Ittoo	ashwin.ittoo@uliege.be	Supervisor Management information systems Natural Language Processing (NLP)
Michael	Schyns	M.schyns@uliege.be	Digital Business Master's coordinator Management information systems Digital Business and Business Analytics
Judicaël	Poumay	Judicael.Poumay@uliege.be	Management information systems

## Chapter 9

# Bibliography and references

# Bibliography

- [Affiah, 2022] Affiah, I. (2022). How to Implement K fold Cross-Validation in Scikit-Learn. <https://www.section.io/engineering-education/how-to-implement-k-fold-cross-validation/>.
- [Aguiar and Waldfogel, 2021] Aguiar, L. and Waldfogel, J. (2021). Platforms, Power, and Promotion: Evidence from Spotify Playlists\*. *The Journal of Industrial Economics*, 69(3):653–691.
- [Atha, 2021] Atha, R. (2021). Building Classification Model with Python. <https://medium.com/analytics-vidhya/building-classification-model-with-python-9bdfc13faa4b>.
- [Berns and Moore, 2012] Berns, G. S. and Moore, S. E. (2012). A neural predictor of cultural popularity. *Journal of Consumer Psychology*, 22(1):154–160.
- [Bhattacharjee et al., 2007] Bhattacharjee, S., Gopal, R. D., Lertwachara, K., Marsden, J. R., and Telang, R. (2007). The Effect of Digital Sharing Technologies on Music Markets: A Survival Analysis of Albums on Ranking Charts. *Management Science*, 53(9):1359–1374.
- [Ceci, 2023a] Ceci, L. (2023a). Top Android apps by global downloads 2023. <https://www.statista.com/statistics/693944/leading-android-apps-worldwide-by-downloads/>.
- [Ceci, 2023b] Ceci, L. (2023b). Top smartphone users activities 2022. <https://www.statista.com/statistics/1337895/top-smartphone-activities/>.
- [Chakor, 2019] Chakor, T. (2019). Rap et médias, de la dépendance à l’indépendance ? <https://ventesrap.fr/rap-et-medias-de-la-dependance-a-lindependance/>.
- [Cross, 2016] Cross, A. (2016). There’s a New Way to Measure Commercial Success in Music and It Involves Music Streaming — Alan Cross. <https://www.ajournalofmusicalthings.com/theres-a-new-way-to-measure-commercial-success-in-music-and-it-involves-streaming/>.
- [Curry, 2023] Curry, D. (2023). Music Streaming App Revenue and Usage Statistics (2023). <https://www.businessofapps.com/data/music-streaming-market/>.
- [Degenhard, 2023] Degenhard, J. (2023). Global: Internet users 2013-2028. <https://www.statista.com/forecasts/1146844/internet-users-in-the-world>.
- [Dertouzos, 2008] Dertouzos, J. N. (2008). Radio Airplay and the Record Industry: An Economic Analysis.
- [Genius, 2020] Genius (2020). 13 Organisé by 13 Organisé. <https://genius.com/albums/13-organise/13-organise>.
- [Genius, 2022] Genius (2022). Pollux - PSLP Lyrics and Tracklist — Genius. <https://genius.com/albums/Pollux/Pslp>.



- [Genius, 2023] Genius (2023). Genius — Song Lyrics & Knowledge. <https://genius.com/>.
- [GitHub, 2023] GitHub (2023). Codeternel/research-thesis: A GitHub repo to keep track of my research thesis advancements and alternative file versions. <https://github.com/Codeternel/research-thesis>.
- [Götting, 2023] Götting, M. C. (2023). Global music streaming subscribers 2022. <https://www.statista.com/statistics/653926/music-streaming-service-subscriber-share/>.
- [Kellaris and Kent, 2023] Kellaris, J. J. and Kent, R. J. (2023). An Exploratory Investigation of Responses Elicited by Music Varying in Tempo, Tonality, and Texture.
- [Le Monde, 2020] Le Monde (2020). La France est-elle vraiment la deuxième terre du rap ? Et si oui, pourquoi ? (Rap Business Ep. 2). <https://www.youtube.com/watch?v=f5kJjAikHbs>.
- [Lee et al., 2003] Lee, J., Boatwright, P., and Kamakura, W. A. (2003). A Bayesian Model for Prelaunch Sales Forecasting of Recorded Music. *Management Science*, 49(2):179–196.
- [Lynne, 2022] Lynne, K. (2022). Around the World — Global Hip-Hop. <https://www.allmusic.com/blog/post/around-the-world-global-hip-hop>.
- [Microsoft, 2023] Microsoft (2023). Visual Studio Code - Code Editing. Redefined. <https://code.visualstudio.com/>.
- [Milkman, 2021] Milkman, S. (2021). Slicing Contemporary Music Tastes by Demographics and Consumption. <https://colemaninsights.com/coleman-insights-blog/slicing-contemporary-music-tastes-by-demographics-and-consumption>.
- [Miller, 2023] Miller, J. (2023). LyricsGenius: A Python client for the Genius.com API — lyricsgenius documentation. <https://lyricsgenius.readthedocs.io/en/master/>.
- [MongoDB, 2023] MongoDB (2023). MongoDB: The Developer Data Platform — MongoDB. <https://www.mongodb.com/>.
- [Mourgere, 2015] Mourgere, I. (2015). Chansons françaises à la radio : Des tubes oui, mais pas que ! — TV5MONDE - Informations. <https://information.tv5monde.com/culture/chansons-francaises-la-radio-des-tubes-oui-mais-pas-que-23705>.
- [Muthukadan, 2023] Muthukadan, B. (2023). Selenium with Python — Selenium Python Bindings 2 documentation. <https://selenium-python.readthedocs.io/index.html>.
- [Oliver, 2020] Oliver, M. (2020). Paris, France Was the Most Successful City for Hip-Hop in 2019. <https://djbooth.net/features/2020-01-09-france-got-something-to-say-album-sales>.
- [Pandas, 2023] Pandas (2023). Pandas - Python Data Analysis Library. <https://pandas.pydata.org/>.
- [Petrosyan, 2023a] Petrosyan, A. (2023a). Devices used to access the internet 2022. <https://www.statista.com/statistics/1289755/internet-access-by-device-worldwide/>.
- [Petrosyan, 2023b] Petrosyan, A. (2023b). Reasons for using the internet worldwide 2022. <https://www.statista.com/statistics/1387375/internet-using-global-reasons/>.
- [Python, 2023] Python (2023). Welcome to Python.org. <https://www.python.org/>.
- [Smirke, 2023] Smirke, R. (2023). IFPI Global Report 2023: Music Revenues Climb 9% to \$26.2 Billion. <https://www.ifpi.org/ifpi-global-music-report-global-recorded-music-revenues-grew-9-in-2022/>.

- [SNEP, 2023a] SNEP (2023a). Les certifications. <https://snepmusique.com/les-certifications/>.
- [SNEP, 2023b] SNEP (2023b). L'organisation. <https://snepmusique.com/lorganisation/>.
- [Spacy, 2023] Spacy (2023). spaCy · Industrial-strength Natural Language Processing in Python. <https://spacy.io/>.
- [StatistaMarketInsights, 2023] StatistaMarketInsights (2023). Music Streaming - United States — Statista Market Forecast. <https://www.statista.com/outlook/dmo/digital-media/digital-music/music-streaming/united-states>.
- [StatistaResearchDepartment, 2023] StatistaResearchDepartment (2023). Global: Mobile app revenue by segment 2019-2027. <https://www.statista.com/forecasts/1262892/mobile-app-revenue-worldwide-by-segment>.
- [Strobl and Tucker, 2000] Strobl, E. A. and Tucker, C. (2000). The Dynamics of Chart Success in the U.K. Pre-Recorded Popular Music Industry.
- [Webster, 2023] Webster, M. (2023). Definition of SUCCESS. <https://www.merriam-webster.com/dictionary/success>.
- [Wikiwand, 2019] Wikiwand (2019). Wikiwand - Syndicat national de l'édition phonographique. [https://www.wikiwand.com/fr/Syndicat\\_national\\_de\\_l'edition\\_phonographique](https://www.wikiwand.com/fr/Syndicat_national_de_l'edition_phonographique).
- [Wikiwand, 2023a] Wikiwand (2023a). Latent Dirichlet allocation - Wikiwand. <https://www.wikiwand.com/en/Latent%20Dirichlet%20allocation>.
- [Wikiwand, 2023b] Wikiwand (2023b). RIAA certification - Wikiwand. [https://www.wikiwand.com/en/RIAA\\_certification](https://www.wikiwand.com/en/RIAA_certification).
- [Wikiwand, 2023c] Wikiwand (2023c). Wikiwand - Disque de certification. [https://www.wikiwand.com/fr/Disque\\_de\\_certification](https://www.wikiwand.com/fr/Disque_de_certification).
- [Wikiwand, 2023d] Wikiwand (2023d). Wikiwand - H.I.P. H.O.P. [https://wikiwand.com/fr/H.I.P.\\_H.O.P](https://wikiwand.com/fr/H.I.P._H.O.P).
- [Wikiwand, 2023e] Wikiwand (2023e). Wikiwand - Skyrock. <https://wikiwand.com/fr/Skyrock>.

# Executive summary

The topic of this master research thesis was to discover what features can explain the commercial success of a French hip-hop released song. To do so, the decision was taken to analyse certified French hip-hop albums, as they gathered many songs together and provided a time frame and other possibilities for correlation. We used a classification model approach to train a model to be able to predict correctly what set of features would reach what certification.

To start, we created a MongoDB database with all of the certified French hip-hop albums that we retrieved from SNEP, the national organisation for music certification. To filter records that were not hip-hop, we used a non-exhaustive, yet as rich as it gets list of the French hiph-hop artists available on Genius. For each list entry, we queried SNEP website to download the corresponding *.csv* file that contained certified albums data.

Having our certified albums list, we proceeded to input these names, one album at a time, into a Python library named LyricsGenius. LyricsGenius is an API that allows to retrieve all data related to specific albums stored in Genius. This is where we got, among other elements, the lyrics of each song.

Once all data related to each song was collected, it was time to run a topic modelling algorithm, which happened to be a Latent Dirichlet Allocation one in our case. After some pre-processing, all of this data was then ready to be stored in MongoDB.

Having our database system storage, we now accessed it through Python, and more precisely through Pandas, a library to display and work with tables (called "dataframes") more easily. Here again, we had some pre-processing to do before model training: changing topic columns' content, variable dummification, topics grouping to decrease model size and improve efficiency, etc. We checked variable independence and then split our data set into train, validation and test sets.

At that time, we tested plenty of different prediction algorithms, such as Naive Bayes, K-nearest neighbors, and so on. It was also at that moment we take a look at the scientific literature to see what additional feature to take into account, learnings that we applied at best. Our metrics highlighted the over-sampled version of the Random Forest algorithm was the best pick for our set of data, with no less than 83% of F1-score, which is the metrics we decided to focus on, as it encompassed both precision and recall.

Having this trained model, we were now able to make predictions. A sample of the result can be found in our appendices. We observed that the quality of our model was confirmed.

At last, we saved our model in a *.joblib* file for reusability. We then proceeded to discuss the biggest problems we encountered in our model, and we described how we fixed each of them.

Although the result of our model was very good, there were many other features to test and hypothesis to try. That is what we discussed very briefly in the limitations of this thesis. We indeed believe that there remains a lot of data to analyse, useful discoveries to make and we tried to give some of these future research paths to conclude this thesis.

**Number of words:  
25,158**