DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2016

Analyzing user behavior and sentiment in music streaming services

AHMED KACHKACH

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

Analyzing user behavior and sentiment in music streaming services

AHMED KACHKACH

Computer Science - Master's programme in Machine Learning
Supervisor: Hedvig Kjellström
Examiner: Danica Kragic
Principal: Anders Arpteg

Published: 04/05/2016

Abstract

These last years, streaming services (for music, podcasts, TV shows and movies) have been under the spotlight by disrupting traditional media consumption platforms. If the technical implications of streaming huge amounts of data are well researched, much remains to be done to analyze the wealth of data collected by these services and exploit it to its full potential in order to improve them.

Using raw data about users' interactions with the music streaming service Spotify, this thesis focuses on three main concepts: streaming context, user attention and the sequential analysis of user actions. We discuss the importance of each of these aspects and propose different statistical and machine learning techniques to model them.

We show how these models can be used to improve streaming services by inferring user sentiment and improving recommender systems, characterizing user sessions, extracting behavioral patterns and providing useful business metrics.

Analysera användares beteende och sentiment i musikströmningstjänster

Sammanfattning

De senaste åren har strömningtjänster (för musik, podcasts, TV-serier och filmer) varit i strålkastarljuset genom att förändra synen på hur vi konsumerar media. Om det tekniska impikationerna av att strömma stora mängder data är väl utforskat finns det mycket kvar i att analysera de stora datamängderna som samlas in för att förstå och förbättra tjänsterna.

Genom att använda rådata om hur användarna interagerar med musiktjänsten Spotify, fokuserar den här uppsatsen på tre huvudkoncept: strömmandets kontext, användares uppmäksamhet samt sekvensiell analys av användares handlingar. Vi diskuterar betydelsen av varje koncept och föreslår en olika statistiska och maskininlärningstekniker för att modellera dem.

Vi visar hur dessa modeller kan användas för att förbättra strömmningstjänster genom att antyda användares sentiment, förbättra rekommendationer, karaktärisera användarsessioner, extrahera betendemönster och ta fram användbar affärsdata.

Acknowledgements

I would like to start by thanking the three supervisors who helped me complete this thesis project:

Thanks to Anders Arpteg, my supervisor at Spotify, for his continuous guidance and for sharing some of his machine learning knowledge.

Thanks to Hedvig Kjellström, my supervisor at KTH, for her valuable advice and for helping me make this report less of a mess.

Thanks to Mehdi Kaytoue, my supervisor at INSA de Lyon, for his supervision, data-mining labs and exciting StarCraft-based papers.

Thanks also to Danica Kragic for examining this thesis. Thanks to my Spotify colleagues Boxun Zhang, Magnus Petersson, Karina Bunyik, Ludvig Fischerström and Martin Håstad for helping with parts of this project, spell-checking and countless fikas.

Finally, to end on a cliché note, none of this would have been possible without the continuous support of my wonderful parents, Abdelaziz Kachkach and Saadia Nadifi, who sacrificed all they had (and more!) to support me and encourage me from my first math classes to my studies abroad.

Contents

Chapter 1

Introduction and background

In this chapter, we introduce the subject of this degree project, give some information about its context and motivations, ethical implications, present the project's methodology and our contributions to the subject.

Streaming services took the entertainment industry (music, films, TV, ...) by surprise, and services like Spotify and Netflix are now more than just distant threats to traditional media providers. This growth is accompanied by an increasing amount of data collected during the use of such services: Media streaming services maintain continuously interact with the users and hence collect more information about their usage behavior, contrary to traditional media distribution (including digital platforms) which have to rely on sales information to infer user preference.

This data can be used to infer a large array of information about users, their behavior on the service, the service's features and the content provided. We will show how it is possible to analyze multiple signals collected from the interaction of the users with the service and build better models of user behavior and understand their habits. By doing so, we can provide useful metrics for user satisfaction and sentiment, and insights about the service that can help improve it or even improve the input given to personalization and recommendation systems.

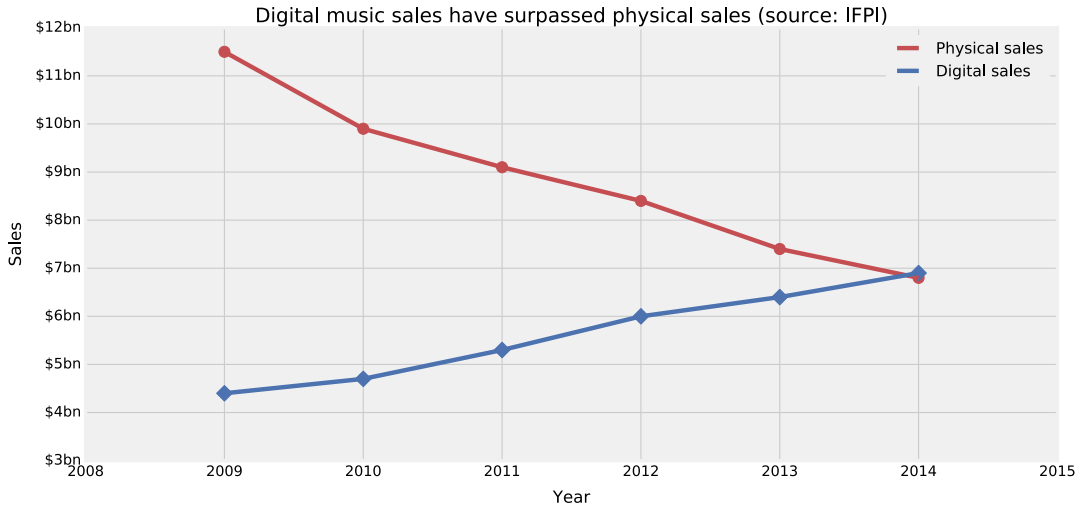
This degree project was conducted with the music streaming service **Spotify**, and uses data provided by the company. However, most of the concepts introduced here are generalizable to any similar service, and some can be applied to in other contexts such as video streaming or news reading.

1.1 Streaming services and the music industry

The music industry is one of the biggest industries in the world, with over 15 billion dollars in revenue in 2015[1]. It has been going through major changes in the last years: a gradual take-over of physical sales by digital downloads (with platforms such as *iTunes*), and the recent success of music streaming services that are seen by everyone as the future of the industry.

"Digital music revenues overtake physical sales for the first time", "Even retail executives think brick-and-mortar-only stores are headed for extinction"... Such are the headlines after the *IFPI*, an organization that represents the interests of the recording industry, published its annual report for 2015. This report contains a number of interesting figures, most notably the fact that digital sales (slightly) surpassed physical sales in 2014.

[1]This figure as well as the ones quoted below are from the IFPI's annual report for 2015: http://www.ifpi.org/downloads/Digital-Music-Report-2015.pdf



But more than the long-predicted take-over of digital media, the most surprising and disruptive evolution that the music industry witnessed is the arrival of legal music streaming services. These services already represent 32% of revenues in digital sales, with a growth of 39% in the number of subscribers in 2014. In some countries, like Sweden and South Korea, streaming already represents more than 90% of digital revenue!

This trend is also present for other types of media: TV, movie tickets and DVD sales are falling, while video streaming services like *Netflix* and *HBO Go* are increasingly popular.

1.2 Spotify

Spotify is a music streaming service with a large catalog, available on multiple platforms (desktop, mobile, game consoles, cars, ...). They are currently the leaders, with over 75 million active users, of which more than 30 million are paying users [2], but are facing new challenges with Apple's and Google's entry into the market.

One of the core components of Spotify's offering is the customization offered to every user: every user gets music recommendations tailored to them by analyzing their listening history. Spotify has recently put an even stronger emphasis on data and aims to be a "data-first company". One of Spotify's recent recommendation features ("Discover Weekly", a weekly playlist built based on users' listening history) has received wide press coverage and quasi-universal praise for its surprising ability to discover new tracks the users like.

This customization offering is developed by the cooperation of multiple teams, based in the Stockholm, New York and Boston offices: giving the "right" recommendations does not only involve building the most sophisticated algorithms but also being able to deploy them on Spotify's scale and catering to business constraints like monitoring user engagement, maintaining a high conversion rate to the premium plans and low churn rates. But ultimately, the goal of all these teams is the same: getting a better understanding of their users, their behaviors and preferences.

The clients used to access the Spotify service (on various platforms: mobile, desktop, TV, ...) do not ask users for their satisfaction nor do they ask them to explicitly rate tracks they liked or disliked. These clients can also be used in very different ways: listening to music while on a hour-long commute, playing music in a café throughout the day or actively browsing through dozens of artists using discovery features provided by Spotify.

[2] These numbers are subject to change. The up-to-date figures can be found on Spotify's press website: https://press.spotify.com/us/about/

This variety and absence of an explicit source of ground-truth when it comes to user satisfaction are among factors that make it vital to build robust ways to analyze the user data collected by Spotify to extract business insights, usage patterns, improve recommendations and find ways to improve the overall user satisfaction.

1.3 Recommendation and personalization

Personalization is a vital component of music streaming services, and even broadly all digital media providers: the time spent by users on the service and their satisfaction with it are the main drivers for revenue on these platforms, hence everything is made to keep the user entertained by making him discover more content to consume.

This personalization can take many shapes: Editorial collections grouping items that would be of interest to a relatively large subset of the user-base or automated algorithms that analyze the service's catalog and/or users' consumption history to provide customized recommendations.

Editorial collections and human music curation

Traditionally, people discovered music through television programs, radio channels or specialized magazines and blogs. Streaming services had strong incentives to reproduce the same experience right in their clients: keeping users entertained and making them discover new artists not only means that they will spend more time on the service, but also that they build an emotional link and have an increased trust in the service, as shown in [15]. The simplest and most common way to provide such curation is by providing hand-made playlists and collections created by content curators employed by streaming services. This has some limitations: it provides biased recommendations, misses on more subtle trends and can be seen as a form of elitism by "tastemakers". But such a form of organic recommendations is also appreciated by many, and even seen as a superior form to algorithmic approach, notably by Apple Music who announced that human curation would be at the core of its service, notably with a large array of editorial playlists and the live radio Beats 1. Spotify has been providing human curated playlists for years, but they heavily rely on user data to build such editorial playlists in contrast with other services that rely more on the intuition tastemakers. Lately, Spotify has put more and more emphasis on completely automated forms of recommendation, based on techniques we detail below, including one of their most praised features: Discover Weekly.

Recommender systems

Recommender systems can have different definitions depending on the level at which we place ourselves (business, engineering or computer science), but one high-level definition is that they are systems that provide personalized recommendations to users based on data we have about them and about the items we wish to recommend. This data can be per-recommendable-item features (like the number of times a user have played a track), properties of the items (like the frequency of words present in a document), personal and demographic information about the user (age, country, ...) or a combination of both.

Recommendation algorithms are often divided into two types:

- **Content-based recommendation:** this method uses the intrinsic properties of items to do recommendations. One example would be recommending similar documents based on the frequencies of words they contain. The *Scribd* service is a real world example of this approach, used to provide customization for their *Netflix for books*.
- **Collaborative filtering:** this method uses a user's consumption history to extract the user's preference and how items related to each-other and hence provide personalized

recommendations to every user. This is by far the most commonly used approach: From *Amazon*'s customized e-commerce portal to *Netflix*'s movie recommendations and the systems we will discuss in this research, like *Spotify*'s music recommendation. Collaborative filtering is itself divided into multiple classes of algorithms: Item-Item/User-User recommendations, matrix factorization machines, etc. ; We will discuss every one of these types in more details in chapter 2.

Collaborative filtering is the method that fits music streaming services' use case the most and yields the best results: all the music people listen to and all the playlists they create can be used to infer a model for musical taste. With that being said, it is also common to use content-based methods to solve the "cold-start" problem: when a new song has just been released, nobody has listened to it and as such it is impossible to use collaborative filtering to recommend it, hence limiting its reach. In such a case, the song's actual content can be used: both metadata and the actual audio data, via analyzing its spectrogram for instance. Although generally less accurate, this allows to make recommendations that are not too far off while we gather information to be able to use the collaborative filtering models.

The choice of recommender system is important, but with like with all machine learning models the data we are feeding these models is vital: "Garbage in, garbage out."

1.4 Data collection in music streaming services

Unlike traditional digital music distribution platforms where music can be listened to in different clients and devices without any feedback to the music provider, music streaming services generally only allow music to be played from their clients. As a result, these companies have a greater control over the user experience, which also means that they collect a multitude of signals about their clients such as the type of music their users consume.

This is particularly interesting in the case of recommendation, as simply acquiring an item does not mean that it has been used, and even less that the user appreciated it. But knowing that the user played it for a certain number of times, in a given context, already gives us a higher confidence in the user's preference. [16]

The type of information that is collected by music streaming services varies (with the type of features they provide to their users and the sophistication of their telemetry and data infrastructure), but here follows a non-exhaustive list of some of the most interesting signals:

- Song playback:
  - Metadata about the played song (artists, genre, etc.)
  - How many times was the song played? Was it skipped? How long was it played?
  - Did the user seek through the song? Were they using shuffle mode?
- Client usage:
  - Which pages did the user view? For how long?
  - Which feature is the user using to play music?
  - Which platform is the user using to play music?
- User properties:
  - Was the user a premium or free user while playing music?
  - How much time have they have spent on the service?

These signals are much more sophisticated than classical metrics that simply count song occurrences and are based on the assumption that all playbacks are equal, regardless of important parameters like the user's context, skipping behavior and platform usage.

Unfortunately, even if these signals open the door to more complex product analysis and user modeling, they are rarely used in all their depth. One reason for this is that many companies, including Spotify, still rely on more traditional metrics (like the number of played songs or the skip ratio of a user) which stood the test of time by proving "good enough" for the most common business tasks like user growth forecasting or to monitor the evolution of the service's popularity.

But more subtle or abstract concepts are harder to analyze with these metrics: If a feature increases the average session length, does that actually mean it made users more engaged with the service or did it just make the clients more cumbersome to use, hence increasing the time needed to play music? Is a high skip ratio the sign of users frustrated with the music they are being recommended, or is it simply more convenient to skip through a playlist than pick songs from the user interface?

1.5 Contributions

We study some of the diverse signals collected by streaming services from their users, more specifically in the case of the music streaming service Spotify, and propose methods to model user sentiment and behavior in a more granular and comprehensive way than traditional analyses. We suggest potential applications for these models, such as building better business metrics and tracking user satisfaction towards the service and the content, improving on traditional methods and metrics which present a series of drawbacks (that we also discuss in this study).

We focus on three main facets of music streaming:

- Leveraging streaming context
- User attention
- A sequential analysis of user actions

For each of these facets, we propose machine learning models that are appropriate along with possible applications for these models.

This research is highly exploratory and as such our goal is not to build a single model for user sentiment but rather to propose multiple models for different facets of user sentiment and behavior. Each of these proposed models present a number of advantages compared to other commonly used methods, but they also have a series of limitations that we will cover.

1.6 Ethical aspects

Our study involves large scale data analysis, including potentially identifying information such as user IDs, songs listened to and timestamps. This wealth of data often comes with great risks and can present concerns for the users' privacy.

As Jules and Tene (2012) [17] put it:

> The harvesting of large data sets and the use of analytics clearly implicate privacy concerns. The tasks of ensuring data security and protecting privacy become harder as information is multiplied and shared ever more widely around the world. Information regarding individuals' health, location, electricity use, and online activity is exposed to scrutiny, raising concerns about profiling, discrimination, exclusion, and loss of control.

Spotify has strict data retention policies and uses anonymized datasets for its pipelines, only using clearly identifying information (such as usernames) when necessary to minimize

the risks of any breach of privacy in its products or abuse of user data. Some of our preprocessing routines (described more in details later in this chapter) require us to have access to a detailed playback context that includes identifying information such as usernames and playlist identifiers. We did not use this information to track specific users and used them instead to segment some types of usage. For example: separating a user listening to a playlist they created from a user listening to a playlist created by any other user.

Anonymizing data is hard. Even when datasets are anonymized by assigning arbitrary IDs and removing any usernames, it is common that potentially identifying information leaks. One of the most popular examples of this is the Netflix Prize, a yearly open competition that awarded 1 million dollars to any team of researchers or amateurs that could give significantly more accurate recommendations than the current state-of-the-art used by Netflix. This competition was ended after 2009 because of privacy concerns: even if the data was anonymized, two researchers from the University of Texas managed in 2007 to uniquely identify users by matching ratings from the data set with an external source (ratings from the film rating website Internet Movie Database). In 2009, this lead to a class action lawsuit from four Netflix users, including a user who had her sexual orientation divulged by this privacy leak. [30]

In that sense, large scale analysis applied on user data, even "anonymized", is more dangerous than it seems: innocuous-looking data can quickly turn to identifying information. If anonymized movie ratings can lead to leaking private information about users' behavior, a large scale analysis on anonymized tweets can predict a user's gender, age, regional origin and even their political orientation. [27]

Another dimension to take into consideration when doing research in collaboration with a company in a very competitive field (such as music streaming) is that data that might seem innocuous from a research perspective can leak valuable information to competitors.

With all this in mind, we performed our data analysis conscientiously and restricted ourselves to the bare minimum needed to analyze user behavior. We are confident that the data published in this report is not sensitive for Spotify users, and the dataset used to do this analysis was stored according to the policies in place at Spotify. Additionally, some data that could be sensitive (such as user retention rates or the platform distribution of users) has been normalized. This was rarely needed in this study, and it is indicated whenever the axes are not up to scale.

1.7 Courses and disciplines useful to this project

Even if the subject of this thesis does not directly concern machine learning or statistics, a large number of courses followed at KTH as part of the Machine Learning master program were instrumental in accomplishing this project.

Machine Learning (DD2431) and Advanced Machine Learning (DD2434) were vital in this endeavor. Not only did they provide theoretical and practical knowledge about machine learning, but they also (indirectly) introduced me to the basics of data analysis, exploration and visualization. The chapter covering user context uses a number of machine learning models, ranging from linear models to ensemble boosting-based models, most of which were introduced in these two classes. These courses also helped us avoid the common pitfalls encountered when blindly using machine learning models (overfitting, training and testing on the same dataset, using the wrong metrics to evaluate models, ..) and helped establish a more rigorous evaluation of the models used. The evaluation of search engines seen in Search Engines and Information Retrieval Systems (DD2476) was also helpful in this sense.

Artificial Intelligence (DD2380) introduced general concepts commonly use to build intelligent systems, and inspired the use of Hidden Markov Models to model latent user states. Pattern Recognition (EN2202) gave us a better a understanding of Hidden Markov Models, and certain of their applications (such as using HMMs' log-likelihood for classification tasks).

In addition to these courses followed at KTH, the data-mining course (by Mehdi Kaytoue and Jean-François Boulicaut) followed at INSA de Lyon was doubly important because it introduced me to clustering methods used in one of the sections of this project and was my first introduction to machine learning as a field, and one of the factors that made me join the machine learning master program at KTH.

1.8 Software and libraries

Every project involving data analysis and machine learning usually involves a large number of libraries, and this project is no exception.

The programming language Python was used throughout this project. This very thesis report was written and exported via the Python-based environment Jupyter Notebook (previously called IPython notebook). This environment made it possible to write this thesis report alongside all the code needed to generate plots and extract the needed information. Although unconventional, this proved to be an extremely efficient way to quickly iterate without any interruption and switching between "writing" and "coding" phases, as the two went hand-by-hand.

The data analysis library `pandas` [21] was extremely useful in loading, manipulating, exploring and visualizing the data used in this project. Custom visualization code based on the `matplotlib` library was also written when more elaborate and customized data visualizations were needed.

The machine learning library `scikit-learn` [25] was used for the machine learning part, along with `xgboost` [6] which provides a more efficient Gradient Boosting Trees implementation and `hmmlearn` which provides an easy-to-use implementation of Hidden Markov Models.

1.9 Outline

We start by a study of literature relevant to the subject in chapter 2. This covers fields such as recommender systems and user modeling.

In chapter 3, we present the data used in this project, the methodology followed to extract it and sampling methods used to reduce it to an exploitable size.

We then present the motivation, methodology and results of our experiments. These contributions are presented over three chapters:

- Streaming context (chapter 4): We analyze the impact of streaming context on a user's behavior, and propose a machine learning model for this context, evaluate the impact of different contextual variables and present a novel method for inferring a user's sentiment from their actions and context.
- User attention (chapter 5): We discuss the importance of user attention for a streaming service, present a series of models appropriate for modeling user attention and propose practical applications for these models.
- Sequential analysis of user actions (chapter 6): We evaluate two sequential models for user actions, use them to gather insights on user behavior on Spotify and propose a series of applications for these models.

General conclusions are presented in chapter 7, along with limitations in the results of this project and suggestions of future work.

Chapter 2

Related work

Plenty has been written about analyzing usage patterns in software and inferring users' affinity in multiple contexts (streaming music, reading documents, shopping...), with a majority of work approaching this problem from the fields of information retrieval, recommendation systems, cognitive science and higher level business analysis.

In this chapter, we give an overview of work related to our project and to relevant fields:

- Research describing the inner working of recommender systems, which we need to know in order to develop a better understanding of what is done in this field to infer user preference and sentiment, especially in the case of implicit feedback.
- Research describing methods to infer user preference in the context of recommender systems' evaluation: what data to use? which metrics? how can user preference be defined?
- Last but not least, research on the issue of modeling user behavior and sentiment with statistical and mathematical models or useful heuristics.

2.1 Personalization and recommender systems

Although the subject of this thesis project is not directly related to recommender systems, it is interesting to look at the research in this field since it touches on a common issue: inferring users' preference (for songs, movies, shopping items, ...).

This field has been very active during the last years, seeing many breakthroughs introduce models with radically improved accuracy, and lately tackling higher-level and more complex issues (like recommendations' diversity, user privacy, etc.)

In [18], a summary of matrix factorization techniques applied to recommender systems is given. More traditional techniques rely on item-item or user-user similarity that only consider one dimension (either users or items) and uses a similarity measure to generate recommendations. In contrast with these methods, Matrix Factorization, with methods such as Singular Value Decomposition (SVD), aims to build latent factors that represent the underlying structure of user preference.

This allows for more accurate recommendations, and also greatly reduces recommender systems' running complexity and memory usage since a low number of latent factors are often enough to faithfully represent the data and generate accurate recommendations. The most popular algorithms to learn such a latent representations are: Alternating Least Squares (ALS) and Stochastic Gradient Descent (SGD).

However, classical SVD is challenging in the case of implicit feedback (which is the most common for music streaming services) so many optimizations have to be added to adapt these methods and obtain good results.

[16] is considered a seminal paper with regards to implicit feedback and the challenges encountered when working with it in the context of recommender systems. It tackles these

challenges with a series of optimizations (to traditional factor models) that are specifically tailored for implicit feedback. This paper defines four unique properties of implicit feedback that prevent us from using traditional models, who are most often built for explicit feedback:

- No negative feedback: it is hard to infer the items that the user did not like from implicit feedback.
- Noisiness: Since implicit feedback usually expressed the fact that a user consumed an item, it does not necessarily mean that the user liked it, and we can only try to infer preference from this.
- Confidence vs Preference: Whereas explicit feedback gives a measure of preference, the value of implicit feedback (the number of interactions between a user and an item) only gives us a measure of confidence. The more implicit feedbacks we have for the same item, the more we are confident that the user likes it.
- Evaluating implicit feedback is hard: Whereas explicit feedback allows for the use of classical evaluation methods (like Mean Squared Error), implicit feedback by its nature forces us to build more elaborate evaluation metrics since it brings more information (many implicit feedbacks for the same item, inter-relation of items, etc.)

The authors then propose a new model for implicit feedback, based on previously researched SVD-based factorization methods.

In [12] Netflix's Chief of Product Innovation and Chief Product Officer present the multiple facets of the recommender systems that made Netflix the giant that it currently is (with more than 65 million members, streaming more than 100 million hours of movies and TV shows per day). Instead of relying on a single "silver bullet" algorithm (which is the case for many streaming services), Netflix provides a series of different algorithms giving different recommendations to fit specific use cases. For instance, "Personalized Video Ranker" (PVR) uses multiple user signals with a mix of popularity to provide a personalized video ranking for specific genres (such as Thriller movies), a "Top-N Video Ranker" is used to produce Top Picks for the users, and "Trending Now" uses both live popularity data and personalization to provide movies and shows that are both noticing an increased popularity and relevant to the user. Our main outtake from this paper is the way Netflix is moving away from explicit feedback, and instead combining a multitude of user signals to model their user's preference, and how they are mixing these signals with item-level properties such as popularity, metadata (cast, synopsis, ...) to provide the best experience for their users.

2.2 Evaluating personalization and recommendation

With the increased popularity of online media services, and the focus on recommender systems, it became vital to build robust methods and metrics to evaluate these systems and provide better personalization that suits what users of such services are looking for. In this sense, the research in this field is interesting as it is ultimately about finding systematic ways to evaluate potential user satisfaction and how automated systems can increase it.

[22] is among a number of papers arguing that accuracy-based metrics should not be the only metrics used for evaluating recommender systems, and that there are other important aspects to take into consideration like the recommendations' similarity and "serendipity". Overall, the goal is to do a robust evaluation of the recommendation systems focusing on what the user wants instead of pure accuracy, which often goes against the end goal of a recommender system. An example that is given is that of a travel recommender system that would only recommend trips that were already done by the user: it would have a perfect accuracy, but the user would not have any use for such an application as his goal is to get recommendations for new places to visit. This is also the case with movies (and music) recommendation where we

should keep in mind that users use the systems to discover new items they would not find by themselves.

This idea is confirmed and developed in [10]. Rather than providing a qualitative argument for moving beyond accuracy metrics like the previous did, this one goes more into details and gives concrete examples of aspects to improve, focusing on "Serendipity" and "Coverage", and giving formal definitions for these two metrics. Unfortunately, the paper's definition of serendipity does not seem to be applicable and scalable as it relies on comparison to a "primitive recommender" to define what an unexpected recommendation is. Finally, the paper argues that a trade-off must be made between accuracy, serendipity and coverage: for instance, coverage will tend to decrease as the accuracy decreases (think of a recommender system only recommending popular items vs a system recommending all the catalog) and an increase in serendipity should lead to an increase in coverage, etc.

Other than the metrics to use, evaluating recommendations requires following a set of rigorous practices. [13] discusses this subject and more specifically discusses which methodologies and algorithms to choose for each type of recommendation tasks, with a focus on offline experiments. The paper also shows how using an improper evaluation metric can lead to making wrong choices. The authors note that there has been much focus on developing new recommendation algorithms which makes it harder to choose which algorithm to use, hence the need for a robust evaluation methodology.

The three main parts of this paper are:

- Describing the different type of recommendation tasks, challenges of evaluating them.
- Proposing a protocol for building an offline experiment: data split, significance tests, ...
- Doing an empirical evaluation of known recommender systems algorithms following the previously defined protocol (both in the scenario where the right metrics are chosen and when irrelevant metrics are used for ranking the algorithms)

This publication gives a great overview of the field and the challenges met when evaluating recommender systems. It also provides a formal description of an evaluation methodology that is now common practice in both academia and the industry.

2.3 Modeling user behavior and sentiment

User modeling is at the junction of multiple disciplines: data engineering, machine learning, psychology, cognitive science and other fields. Our primary goal is to build simple and robust models of users' behavior in music streaming services, and that comprises things like the actions user take on such services, how much time they spend in every music session and how we can infer the way they feel about the service or a specific song from their behavior. This is a simple task when users provide an explicit answer (via a satisfaction poll, or rating songs), but that is rarely the case and is also sensitive to multiple biases.

When there is no explicit rating given by users, simplistic signals, such as the number of times content was accessed or the click-through rate (CTR) are used. In [14], the authors argue that these signals are not sufficient to capture the sentiment of the user towards the content after it started being consumed and proposes a new approach: using the time the user spends on a content item (here called "dwell time") as an important metric to determine engagement with the content and predict the user's satisfaction. The paper also proposes a way to normalize this metric over different contexts (like different devices) to remove any bias unrelated to the user, and shows that using this metric on both Learning-To-Rank and Collaborative Filtering tasks yields better results than the currently used click-optimized methods.

Although this paper works on newspapers consumption (on Yahoo's news portal), this paper perfectly fits our use-case of media and music playback, since it shares many of the

important features to use this method: "dwell time" can be either the time spent listening to a song or the duration of a listening session, which similarly gives us information about how satisfied users are with a song (intuitively, listening to 30 seconds of a song then skipping is probably a bad sign) or with the service as a whole. But dwell time normalization is not necessary in our case because we already know the total duration of a song, although analyzing how dwell time varies among different platforms would definitely be of interest.

When using collaborative filtering, the proposed method only yields slightly superior results than the click-based one, and suggests that this might be because dwell time alone is not enough to represent user experience, meaning that we should use additional information at our disposal (context of media playback, events that initiated playback) if we want to have bigger performance improvements.

One way to extract this context is to look at per-session data. This is done in [11] where HTTP transactions with the YouTube servers were collected on a university network and analyze to extract insights about the consumption of such streaming services. Although this paper puts a higher emphasis on the networking side of things, it provides valuable insights into the properties of streaming services, notably the difference between active events, triggered by the user, and passive events (such as videos or songs loaded automatically). This also shows us that analyzing streaming data on a session level gives valuable insights that might not be found by simply looking at a global aggregation of all the streaming data.

A similar but more thorough study is [31], this time using a massive dataset collected from the Spotify service between 2010 and 2011. This paper, similarly to [11], studies session length and its variance among users, but it also focuses on different aspects of user behavior: session and stream arrival patterns, preferred times for streaming music and changing behavior between multiple or single devices and behavior related to device-switching. Mainly, this paper shows that even if users behave in complex ways on such streaming services, it is possible to get more than decent results with enough modeling efforts: for instance, if streams are not uniformly distributed throughout the day, they can be modeled by a non-homogeneous Poisson process.

In contrast to the previously described methods that focused on specific properties of streaming online services, several researches chose to revert the problem of user modeling to a regression task:

Is it possible to map a user's implicit feedback to an explicit rating representing the user's preference towards an item? Would that be preferable to using implicit feedback with implicit-feedback-based recommendation algorithms?

A certain number of challenges and shortcomings of automatically collected implicit feedback were introduced by Hu et al. [16], and make it harder to use such information as a proxy for user preference or an input to a recommender system. In [23], the authors try to counter each of these supposed shortcomings, mainly by showing that there is a correlation between implicit and explicit feedback, and that if implicit feedback is noisy, so is explicit feedback. But the major part of the paper is dedicated to showing that user preference can only be extracted from implicit feedback by finding the correct mapping from implicit to explicit feedback. A linear model capable of finding such a mapping is proposed and shown to give better performance when used for recommendation tasks.

[24] further improves on this idea by comparing the recommendations obtained on two datasets (collected from the music scrobbling service last.fm) using a novel logistic regression model and the state of the art implicit-feedback algorithm introduced in Hu et al. [16]

Chapter 3

Data collection, processing and exploration

Our study heavily relies on the thorough analysis of data collected from Spotify. This chapter describes the dataset we worked with, the methods we followed to extract it, and some general observations about the data.

3.1 Spotify's data infrastructure

Spotify collects a significant amount of data on a daily basis: an average of 14 TB of user/service-related log data per day, which expands to up to 140 TB. Traditional data storage and analysis methods do not scale to such proportions, which is why the analytics teams at Spotify rely heavily on big data systems like the Apache Hadoop framework with its distributed filesystem (HDFS) and cluster computation frameworks like MapReduce and Spark.

Spotify has a Hadoop cluster consisting of 2000+ nodes for a total of 100 PB of storage capacity and 100 TB of RAM. This cluster is used for production pipelines providing features in the Spotify clients (like "Discover Weekly") but is also used for more ad-hoc analysis. Most tasks described below were executed on this cluster using Apache Spark.

3.2 Datasets of interest

For the needs of this research, we will mainly use a set of playback events collected from Spotify clients on all platforms. These playback events, internally called `EndSong`s, contain a great deal of information about the context of playback: the time of playback, the platform used when listening to music, the length of the playback, the feature used, etc. We also used joined these events to two other datasets to gather information about artists and about user sessions. We will describe the fields available in this dataset later in this report.

Because of the large amount of data stored in this dataset (reaching multiple terabytes of `EndSong`s per day in some occasions) we resort to taking a sample from this data. Given the size of the data and the peculiar distribution of streaming data, a robust sampling method has to be designed to gather as much of representative dataset we can while keeping the size of the data small enough to allow for analysis.

In addition to this playback data, we also extract data separating streams into sessions, and historical data about the type of plans the user subscribed to (free, premium, trial, ...) and their activity on the service. These datasets are orders of magnitude smaller than the `EndSong` data, so extracting it is straightforward.

3.3 Sampling

Uniform, per-track or per-user sampling?

Three options are viable given the structure of our datasets:

The simplest method would be to uniformly sample streams by using their ID, taking random playbacks regardless of the stream's user or track. This would make most sense if we were doing a system analysis focused on the stream of data, regardless of context, as it gives a very general view of what is happening on a service-level. Unfortunately, this is not appropriate for our use-case because this produces (given the same sampling size) sparser data covering more users but with only some examples for every user.

Another approach would be to uniformly pick tracks and analyze their streams. This would be an appropriate approach if we wanted to analyze sentiment on a track-level: how well people react to a given track, how the users' behavior for this track is different from other tracks, etc. ; The problem is that this gives less information per-user, and we lose the context associated with every user's bias and characteristics: some users might tend to skip more often than others in all songs they play for instance.

Hence, the most appropriate sampling method is to sample per-user: we sample a certain proportion of Spotify's users and gather all their streams (on a given time period). This produces a denser dataset with more playbacks for every user, which in turns allows us to produce robust models for users' behavior.

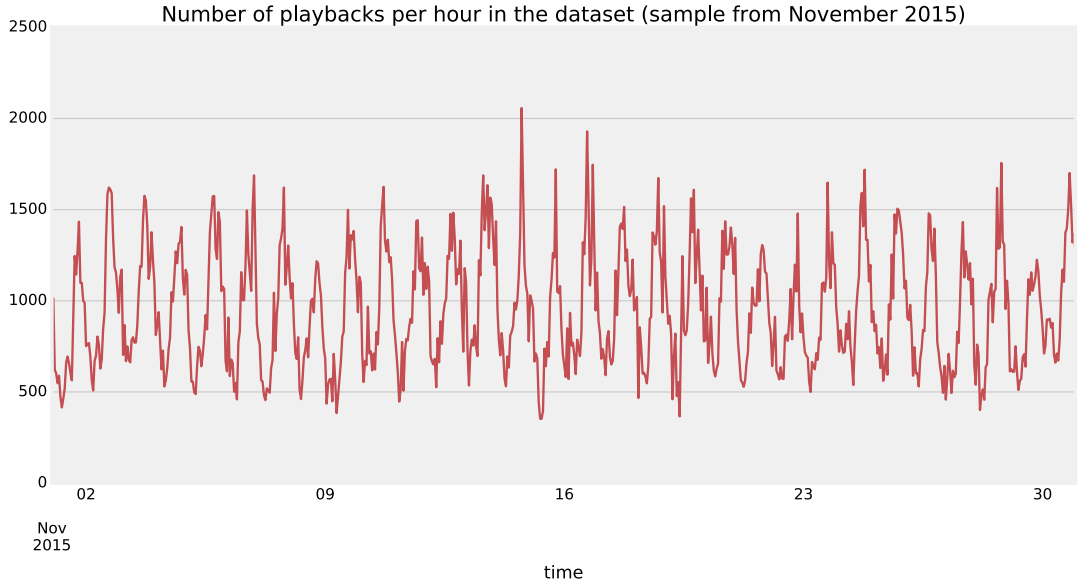
Time and seasonality

Like all online services, music streaming is intrinsically seasonal: most people listen to music during the day than at late hours of night, and although the Spotify service is available worldwide (more than 59 countries) the imbalance in the number of users between different timezone causes the data to remain seasonal.

It is important to note that the time stored at every event is in UTC (Coordinated Universal Time) as timestamps are calculated and stored server-side. Relying on client-side time synchronization is risky as a misconfigured clock might give wrong data and opens the door for many possibilities of abuse. Nonetheless, it is possible to retrieve the local time by using IP location in combination with the server-side timestamp, but for the needs of our research we can simply focus on using per-country analysis whenever time is an important factor (as most countries using Spotify has one timezone or a very small variation of timezones).

Listening behavior can vary wildly depending on the time of the day: people might listen to more or less music during weekdays compared to the weekend, and certain shops broadcasting music via Spotify may only stream during work hours.

Sampling streams uniformly through the data would be biased: We would mostly collect streams from periods where there is a high activity on the service, with a bias towards users that stream more songs, and we would not have enough examples of less dense time ranges. Similarly, collecting streams from the same day of the week, a Saturday for instance, would be problematic as we would suffer from the heavy seasonality effects present in such services and would miss patterns that may only be present in other days of the week.



Sampling protocol

Taking into consideration all the parameters described above, we decided to use the following sampling protocol: We collect data through a whole month. We decided to choose November 2015 for multiple reasons: December has some "uncommon" behavior since it is a holiday period, and earlier months had some data quality issues. We sample per-user to avoid the drawbacks mentionned above.

Sampling implementation and result

Since we are working on very big datasets that we have to join while preserving our sampling constraints, we cannot afford to pass around a user subset to all the executors in the computation cluster at every operation. We chose to instead use a deterministic sampling method: We hash all user IDs and take the wanted proportion of users from these hashes by using the modulo operator. This method allows us to parallelize the data extraction process.

The three main datasets extracted (streams, track data and session data) were joined by using the Spark framework.

The result is a consolidated CSV file containing 740534 playbacks from 1644 unique users.

3.4 Preprocessing

Collecting live data from clients is tricky, and it is common that some logs would be corrupted (either by a bug in the data ingestion pipeline, or by punctual incidents that set incorrect values in some fields). Because of this, we had to preprocess the raw data collected to fix some inconsistencies, and remove instances where information could not be recovered.

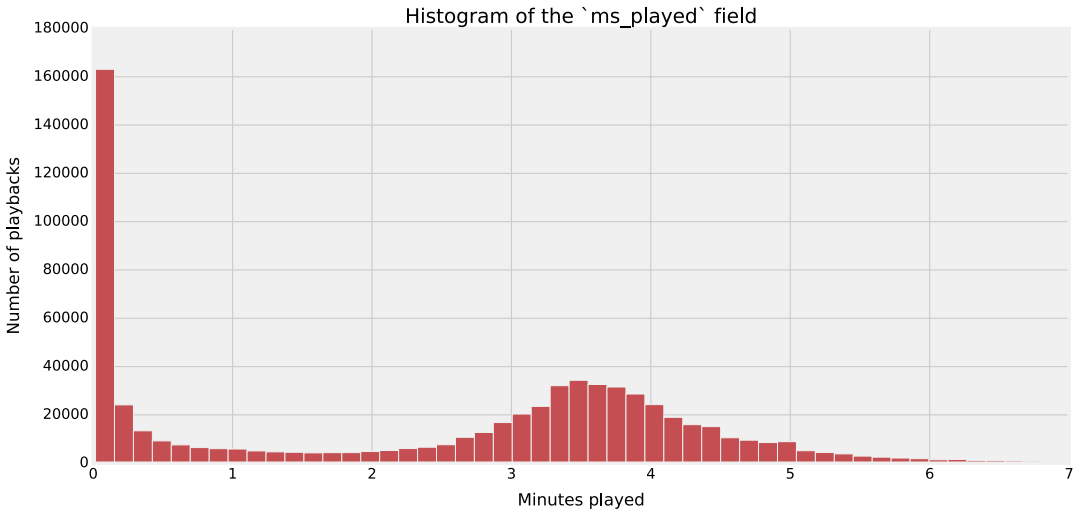
We also added some fields that would make our analysis further. For instance: parsing platform strings to distinguish a smaller number of high-level types of platforms (desktop, mobile and tablet).

The final result is 684807 playbacks with 38 fields each.

Playback length normalization

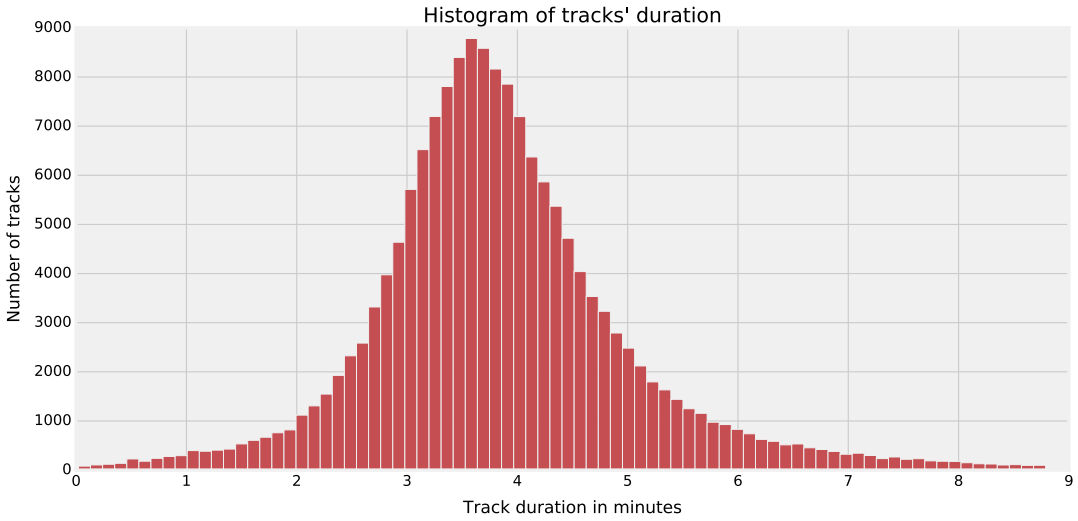
Every stream contains the number of milliseconds played by the user before the end of the playback. This is a very valuable metric since not all playbacks are equal, and playing the entirety of a song is very different from skipping a song after a couple seconds.

This can be confirmed by looking at the histogram of `ms_played` in our sample dataset:



From the figure above, we observe a bimodal distribution between the two ends of the spectrum: a large portion of playbacks have a very short duration (less than 20 seconds) and an even higher portion of playbacks following a Gaussian distribution centered on a 3.8 minutes length. There's also a decent number of streams between these two peaks.

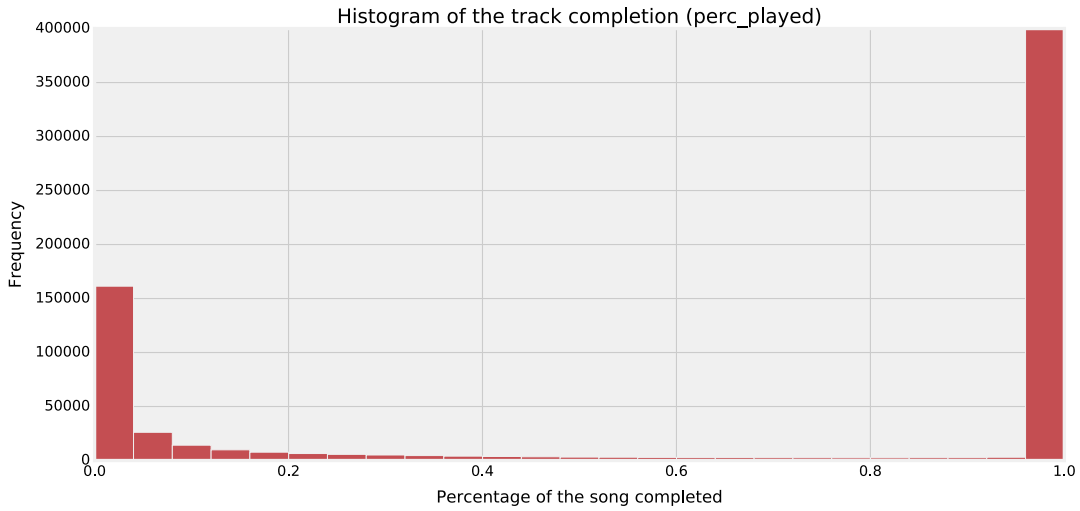
While this histogram is informative, it is also heavily biased by one additional dimension: track length.



The figure above shows that track duration follows a similar Gaussian distribution to the one we noticed on the higher end of the spectrum of stream duration. This is because most of those streams are tracks that were streamed to completion, but vary in their length.

The effect of this variance in track duration is that the length of streams is biased by the duration of the track, and it is hard to differentiate short streams that completed entirely from longer streams that were stopped in the middle.

If the total time spent on a piece of content has an upper limit in music streaming, this is not the case in many web services such as news websites (like Yahoo! news). In the absence of such upper limits, more elaborate processing and normalization has to be done by analyzing user behavior [14], but in our case we can simply normalize every stream by the track duration of its track.



This shows that the playback behavior is more binary than what we would've thought by observing the first histogram. More than 82.56% of streams are either completed entirely or skipped after a very short duration (less than 5.00% of the track's duration).

This normalization is particularly important when we want to compare user behavior among different songs of various lengths. Because of this, we will focus on this measure in the rest of this research, but we will still use the absolute playback duration (`ms_played`) in cases where behavior is invariant with respect to the duration of the track.

Chapter 4

Streaming context

Not so long ago, music consumption was restricted to a limited number of contexts: people would gather around a HiFi system, a radio or a phonograph and listen to the latest musical hits. The first Sony Walkman revolutionized this by making it possible to listen to music privately and in any place.

Nowadays, with the massive penetration of smartphones in all markets (including developing countries) and the increased accessibility and affordability of data plans, this ubiquity of contexts is ever more present, and music is consumed in radically different contexts and conditions. The users using music streaming services are also pretty diverse, with different ways to interact with music and sometimes mechanical limitations related to the type of device, platform or subscription plan they use.

It is hence important to evaluate the impact of context on user behavior. In this chapter, we will study this impact by evaluating how different contextual information impact the nature of streams. We will also present methods and models for this contextual information and possible applications for these models.

4.1 Analyzing streaming context

Platform used: mobile, desktop or tablet

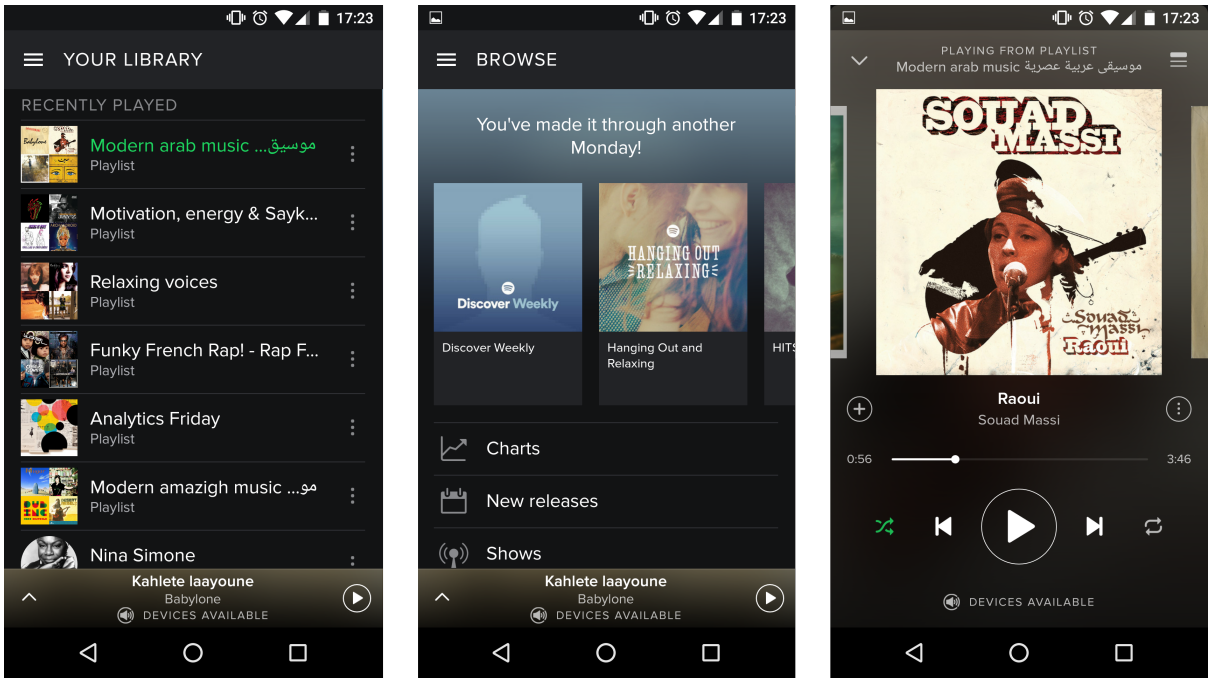
Spotify is present on a multitude of platforms. We can separate them into the following categories:

- **Desktop**: via native apps on Windows, Linux and Mac OS, or with a webplayer straight from any browser.
- **Mobile**: via native apps on iOS, Android and Windows Mobile.
- **Tablet**: via the tablet counter part of the mobile platforms (iPad, Android tablets, Microsoft Surface, . . .)
- **TV, Hi-Fi systems, cars and other embedded systems**: via a set of partnerships with constructors, notably Sony who chose Spotify as the sole music provider on the PS4 game console (more than 36 million PS4s sold since its launch) or the Tesla car.

Although they are promising platforms, Hi-Fi systems, TVs and embedded devices are not included in our dataset, so we will focus on the first 3 types of platform (which represent the majority of the streams).

Even if there exists small differences between clients used on the same platform type (e.g Android versus iOS), the experience tends to be homogeneous among the same device type. This is however not the case for clients on two different types of platforms, say mobile versus desktop, which are radically different since every platform has its specificities and is used in a different context.

Spotify desktop client on Mac OS

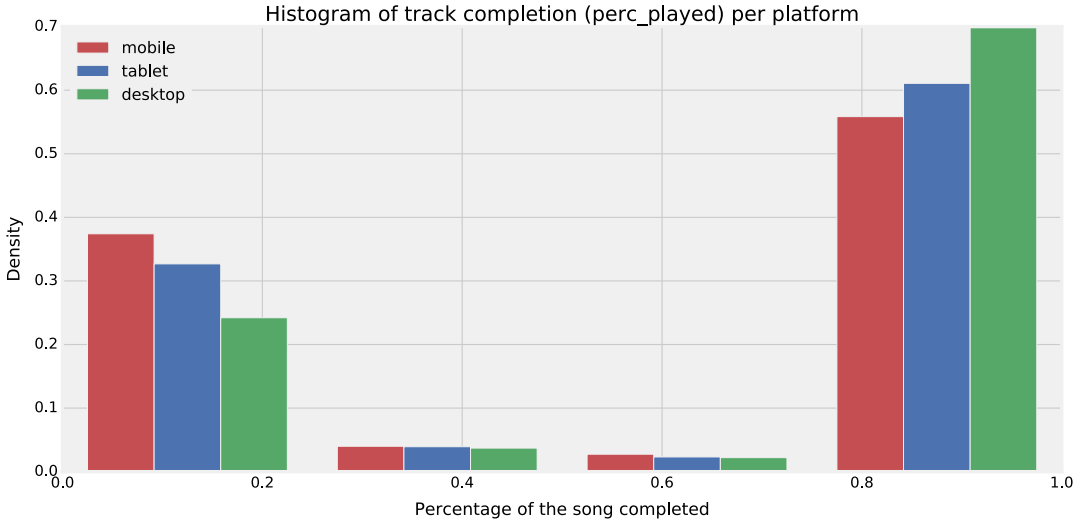


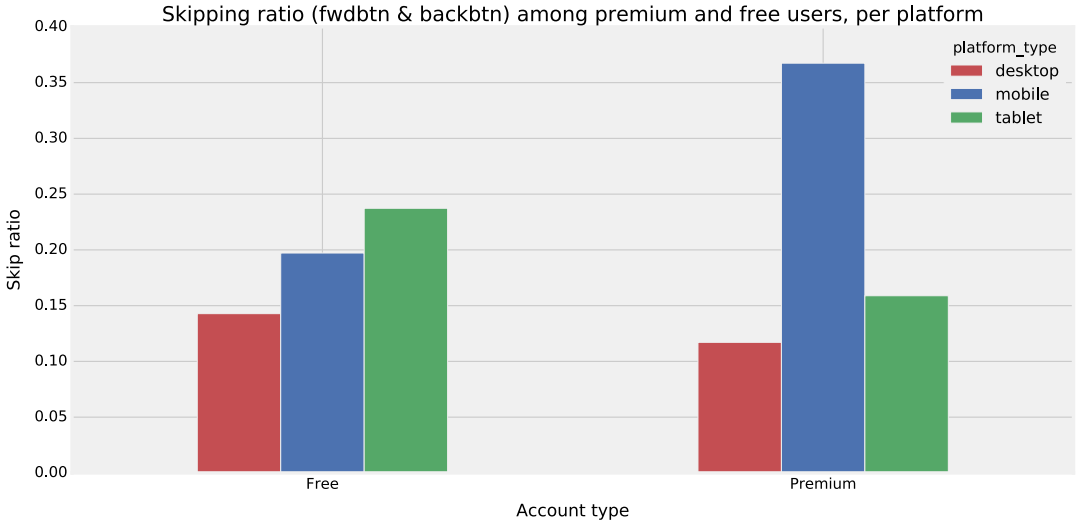
Spotify mobile client on Android

There are some clear differences between the user interfaces on desktop and mobile: skip/play buttons are more prominent on the mobile application, the desktop application shows more elements at once (due to a significantly larger screen size) and some advanced features (like songs your friends are listening to) are missing on the mobile application.

There are also more fundamental differences: mobile applications are used in more nomadic contexts, such as commuting to work or running. Traditional computers (and tablets) are more commonly used in a stationary state, and often left playing music in the background. As such, the clients are optimized for different usages and have different mechanics such as the recently introduced "Running" feature that provides music that adapts the BPM (number of Beats Per Minute in the music) to the running rhythm.

Do these differences translate to difference in behaviors on these platforms? We can confirm this intuition by looking at patterns of behavior on the different platforms:





These graphs show that the platform used for streaming music has a significant impact on the user's behavior.

Moreover, the last graph shows that this difference is significantly bigger in the case of premium users, as they do not have any limitations on skips. We discuss this with more details in the next section.

Mobile premium users are more than 3 times as likely to skip a track than desktop premium users, and more than twice as likely to do so as tablet users.

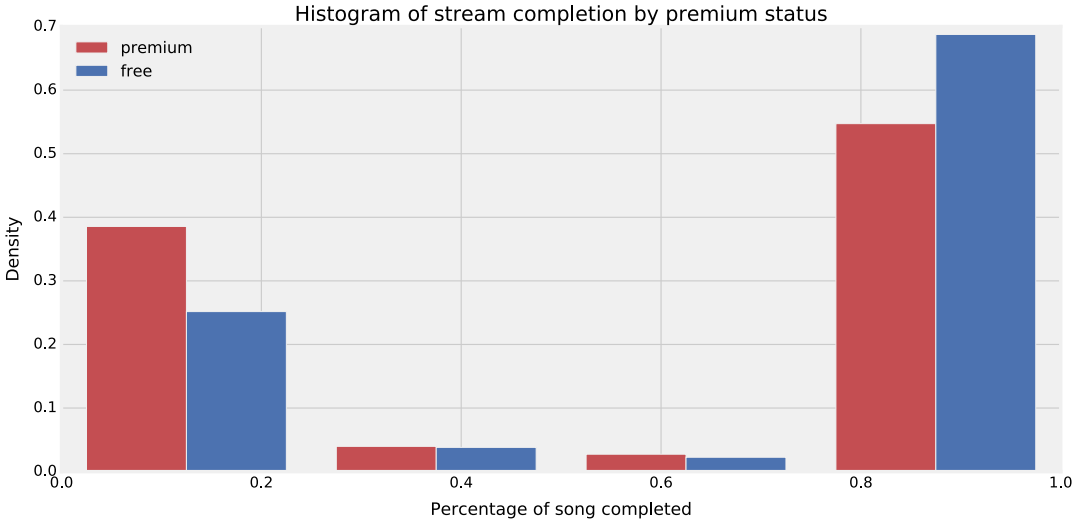
This shows that desktop use of the service tends to be more passive: like playing music during a party or while working, whereas mobile use is more likely to be active. Tablets are between these two worlds, which is intuitive since they are made to be used both in a mobile context and as traditional desktop machines.

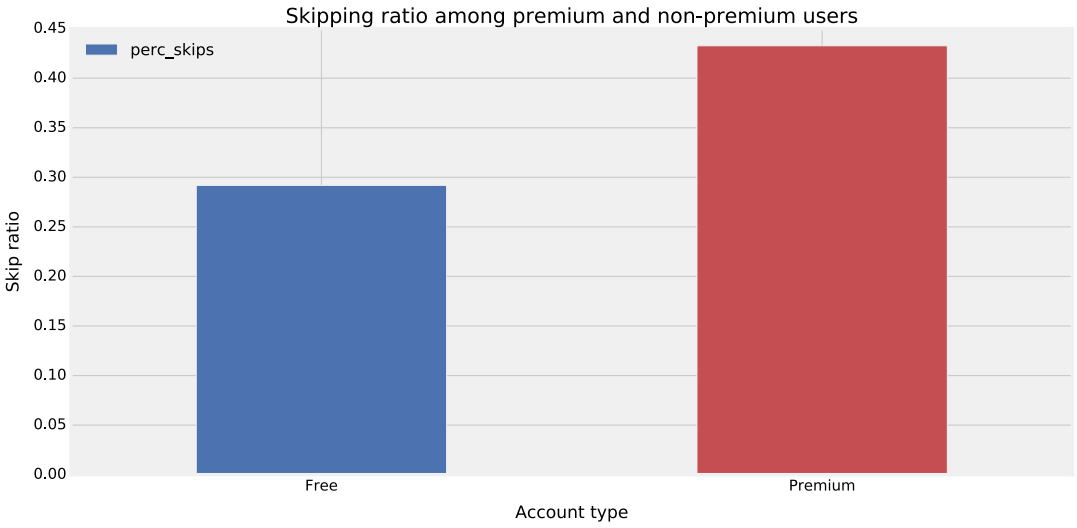
We will get back to analyzing the concept of active and passive listening with more details in chapter 5.

Premium and Free subscription plans

Unlike many other music streaming services (like Apple Music and Google Play Music), Spotify provides a free plan that lets user listen to music without subscribing for their premium plan. The trade-off is that users have to listen to or watch an advertisement after a certain number of streams. In addition to this, there are also some mechanical limitations for users of the free plan: they cannot listen to music offline and mobile users of the free plan have a limited number of streams.

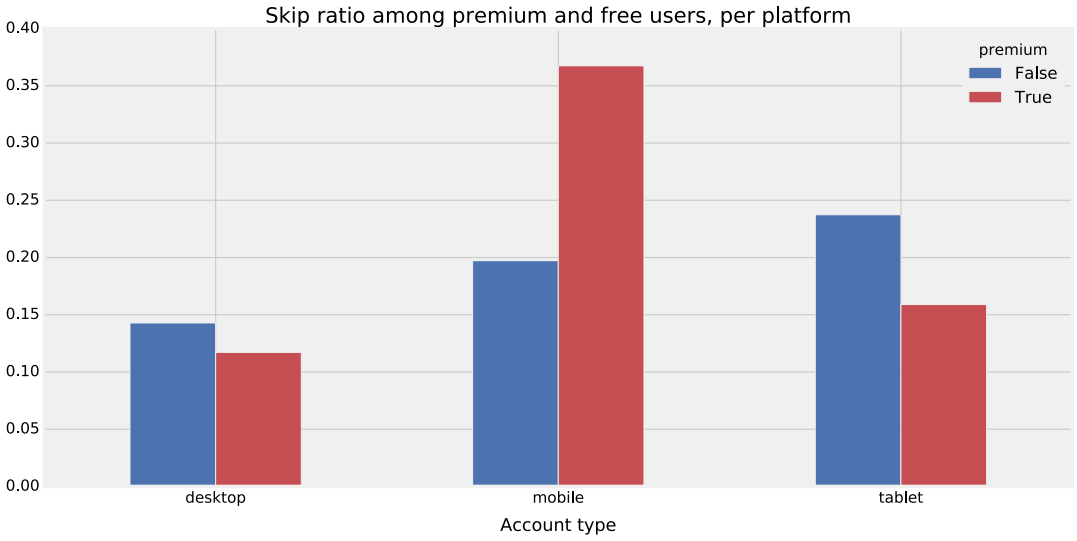
As expected, these differences have a deep impact on user behavior:





The above graphs show a clear difference in behavior between users of the premium and free plans: Premium users skip tracks significantly more often than free users.

Like shown in the previous section, this difference is even more noticeable when we look at the effect a premium plan has on every platform:



Why is there such a big difference, especially in the case of mobile clients? Two main reasons that we previously touched on:

- The fact that free users on desktop **have to listen to ads** after a certain number of tracks played might condition them to skip less.
- In addition to **ads**, free users on mobile also have a **limited number of skips (6 per hour)** which mechanically restricts the number of skips in this population.

Additionally, people getting premium accounts do not have the same demographics as users of the free plan (for instance, students and younger users tend to use the free plan more

than other demographics), and users that are passionate enough about music to take a subscription plan are likely to exhibit different behaviors when it comes to music streaming.

Therefore, it is clear that certain combinations of subscription plan and platform are biased towards more or less skipping, since it can have a higher cost in some cases, and hence both the platform type and whether the user is premium or not should be considered when modeling the user's sentiment and analyzing user behavior.

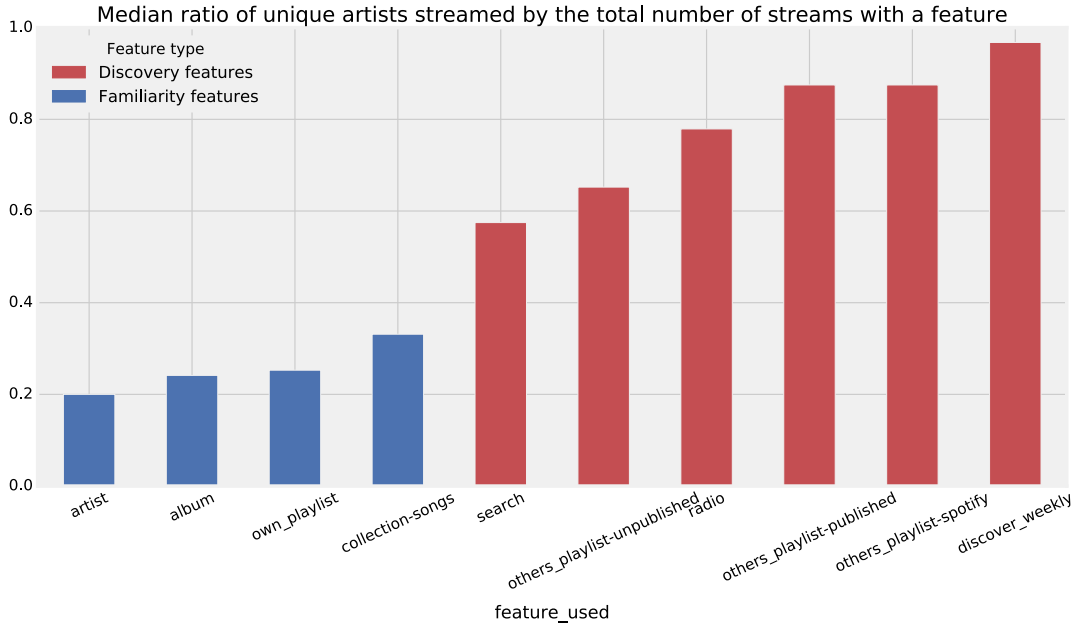
Features used to stream music

Spotify users can stream music through a multitude of features: playlists, interactive radio mode or the recently introduced "Discover Weekly" feature, which gives user a weekly dose of music recommendation based on the millions of playlists created by other users and the user's streaming history.

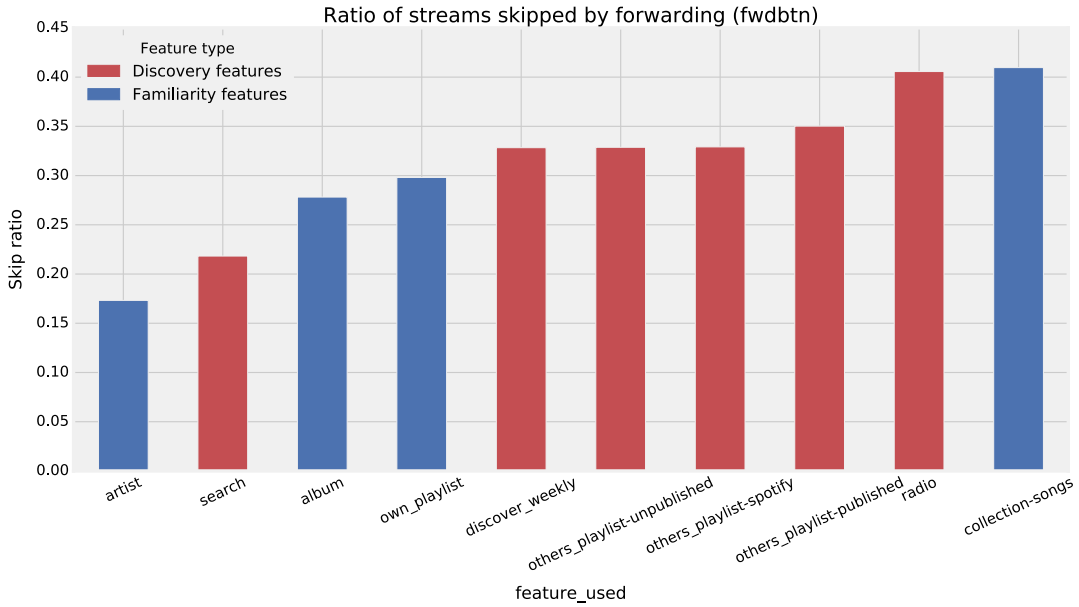
These features can be either aimed towards more discovery (like the "radio" mode and "Discover Weekly") or provide more familiar songs (like the user's saved songs, or curated playlists provided by Spotify). This will naturally bias users towards certain types of behavior more than others (skipping more or less often, listening to different artists or focusing on a small number of familiar artists, etc.)

The graphs that follow explore some of these differences. For reference, here is a short description of the most common values of the `feature_used` field used below:

- `artist`: Streaming a track from an artist's profile page.
- `album`: Streaming a track from an album.
- `own_playlist`: Streaming a track from a playlist created by the user himself.
- `collection`-songs: Streaming a track from the user's collection (where the user saves his favorite songs)
- `search`: Streaming a track found through the search bar.
- `others_playlist-unpublished/published`: Streaming a track from a playlist from some user's playlist.
- `others_playlist-spotify`: Streaming a track from a playlist created by Spotify.
- `radio`: Streaming a song from radio mode, a mode that continuously suggests songs similar to a specific song or genre.
- `discover_weekly`: Streaming tracks from the Discover Weekly playlist, a new feature introduced by Spotify. Gives users a weekly dose of recommendations.



The above graph gives an idea of how much every one of these features is geared towards music discovery: we count the number of unique artists listened to using a certain feature, and it by the total number of streams completed using this same feature. Naturally, discovery-oriented features like Discover Weekly and the radio mode are ranked high, and songs and playlists collected by the user have a low discovery ratio. But we are also have some unexpected results: playlists created by others (including Spotify) have a high discovery ratio, even slightly superior to that of the radio mode. One possible explanation is that users use these features for a more active discovery: going through these playlists once or twice, and saving the songs they like to their collection or personal playlists, where they go to listen to more familiar artists.

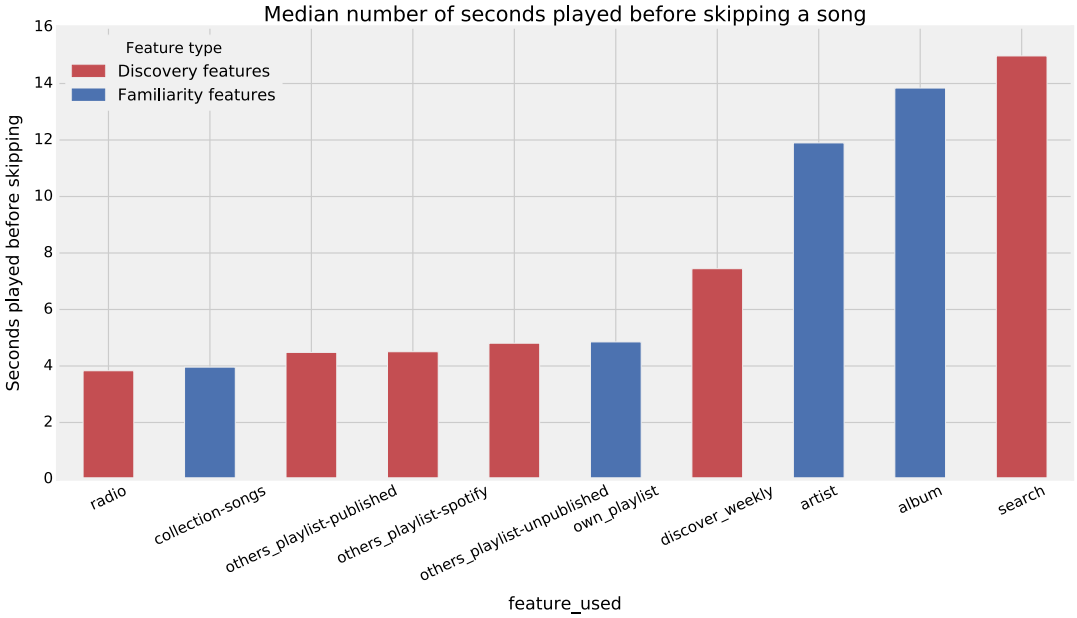


Like said previously, skipping is an event that can carry multiple meanings: it can be used as a way to quickly sift through a playlist looking for a song, or when the user is oriented towards music discovery and dismisses songs he does not like until he finds something to his liking.

This is confirmed by the above graph which shows the skip ratio per feature, divided using the classification we took from the previous graph.

We notice that discovery features have a higher skip ratio than features oriented towards streaming familiar songs. This is something we would expect as users are more likely to find songs they like in the second class of features. There are however two exceptions to this:

- Songs in a user's collection have a high skip ratio: this is explained by the fact that the user saves songs he likes in this collection, and this makes it more likely for him to skip a certain number of times this collection looking for a specific song he saved. This is corroborated by the predominance of very short skips in this feature.
- Search, a discovery feature, has a low skip ratio: This is easily explained by the fact that songs started from search, unlike other discovery features, are chosen by the user (through a search query), and hence have a lower chance of being skipped. In this sense, this feature works similarly to the "artist" and "album" features.



The graph above does not follow the previously introduced classification (discovery vs familiarity) but instead shows a new distinction: skips done in most features arrive after a very short period (around 4 seconds), while other features have skips that arrive significantly later (with a median between 12 and 16 seconds).

One possible explanation is that the later class of features favor listening to songs picked by the user themselves. This is true for search, even if it a discovery feature (as said previously in the skip ratio analysis). A shorter skip have two interpretations: skipping a song that the user played previously to listen to something new (as it is likely the case in `own_playlist` and `collection-songs`) or skipping songs that have a different style from what the user likes (like it might be the case in `radio`). In both cases, the features with a low skip duration are features where the songs are often not picked by the user, but part of a larger collection.

We can note that Discover Weekly, even if it should be long to the first class, has a significantly higher skip duration. This might mean that the recommendations given in this feature are more appropriate than those given in the similar radio mode.

Events starting and ending a stream

A stream can be started or stopped by many different ways, each having their own characteristics. For example, clicking on a song in an artist's page is different from skipping through multiple songs in the currently playing playlist.

This information is encoded in two fields in our dataset: `reason_start` and `reason_end`, which are quasi-symmetrical in the sense where `reason_start` for a stream can often be related to `reason_end` of the previous stream. We will look into this transitional nature in chapter 6, but for now we will analyze the user behavior for every one of these actions.

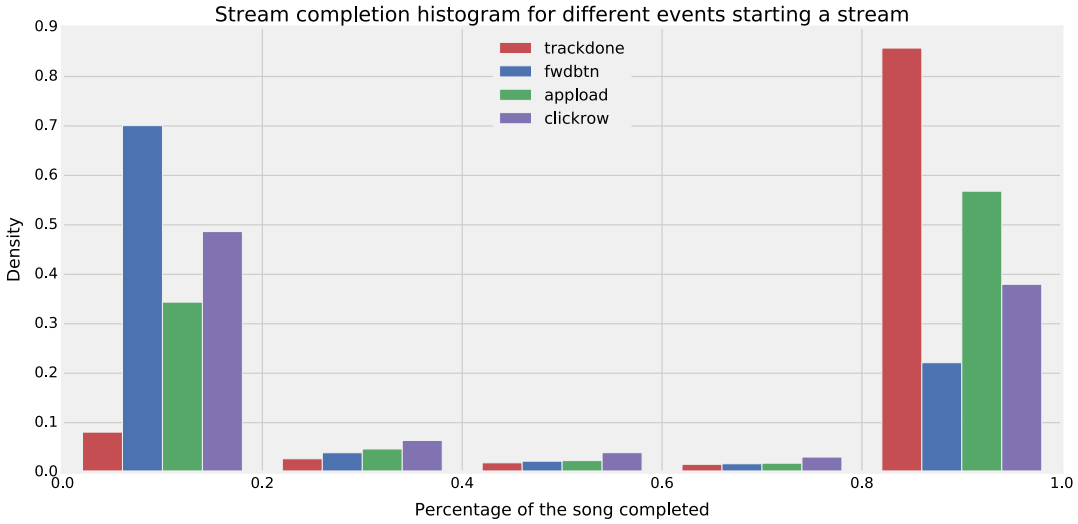
The most common values for `reason_start`:

- `appload`: the stream started when a Spotify client was opened.
- `fwdbtn` and `backbtn`: the stream was started after a previous song was skipped using the forward or back button. Given the similarity of these two actions, we consider both as being `fwdbtn` in the rest of this analysis.

- `clickrow`: the stream started after the user clicked on a list element (a song in a playlist, an artist's page or an album)
- `trackdone`: the stream started after a previous song was completed.

And similarly for `reason_start`.

What effect do these starting events have on the stream itself? Let's use the same type of analyzes we did in previous sections to see if there's a significant effect:



The starting event has a significant effect on the streams' length. For instance, a stream started right after a stream was completed (`reason_start = trackdone`) will be entirely completed around 85% of the time, whereas a stream that was started after a song was skipped with the forward or back button (`reason_start = fwdbtn`) will only be completed 20% of the time.

It is also interesting to notice that skipping with the forward/back buttons is different from skipping by clicking on items from a list (`reason_start = clickrow`). The latter is more conscious (as the user chooses which song to play) and naturally induces a higher chance of completing a song (about 38%, almost double that of a skipped song).

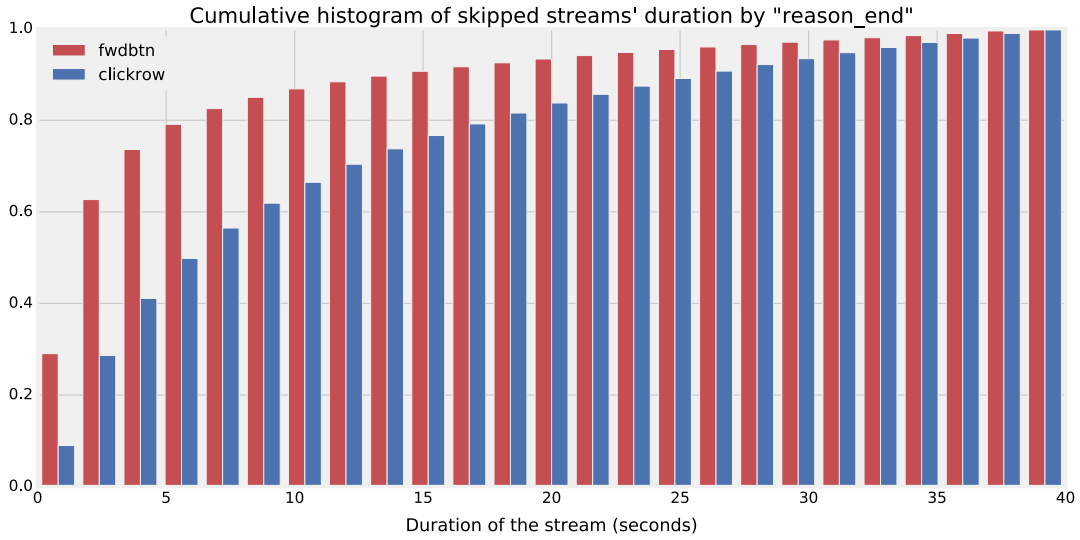
We extract two main results from this analysis:

- The impact of user actions is significantly higher than that of other parameters (like platform type, account type) on the overall behavior of a user. This means that there is a significant loss of information when analyzing user behavior over a long period of time (a month, like we did previously) and amalgamating different states the user might be in.
- Some actions seem to make the user go into a quasi-stationary states: Ending a stream with `fwdbtn` means the next stream has a 80% chance of being skipped as well. And the same is true for completing songs after a `trackdone`.

Both results point to the importance of doing a stream-by-stream analysis in contrast with using aggregations over long periods of time, which miss on important signals related to user behavior.

One way to do such analysis is to build a statistical model of how users interact with clients, and generate a sequence of events. We will explore this idea, and the available options to build such a model, in a later chapter.

An additional angle for analyzing these events is to take into consideration the way they are generated: active events (like `fwdbtn` and `clickrow`) are generated by a click on the interface or on dedicated buttons on the user's device. The user has to move his mouse (or index) over a list to pick a song and generate a `clickrow` event, but he can simply double-tab his volume button on some mobile devices, or click the forward button on a keyboard, to move to the next song and generate a `fwdbtn`. Because of this, playbacks ended by such events will have significantly different durations:



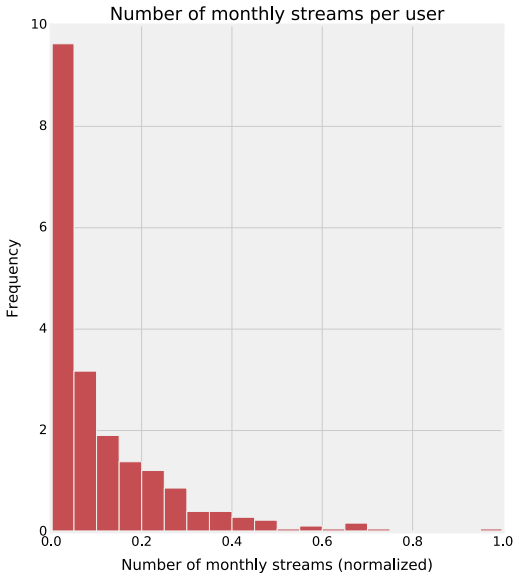
There are two main insights we can extract from this:

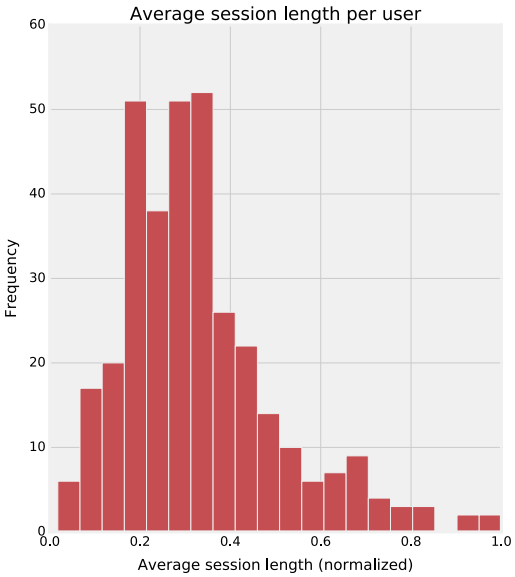
- 80% of streams skipped with `fwdbtn` were skipped after less than 5 seconds! This shows that quickly skipping through playlists is a very common habit among users: this confirms that not all skips are necessarily a negative signal for the skipped songs.
- Skips with `fwdbtn` happen significantly faster than skips with `clickrow`: Users usually take some time to listen to a song they just selected (as can be seen with the previous graph analyzing `reason_start`). This might mean that users skipping a song with a `clickrow` are more likely to have disliked it, since they took some time to listen to it and were not just sifting through a playlist looking for a song. This shows that not all skips are equal, and the feature used to skip a song has an impact on the meaning it carries.

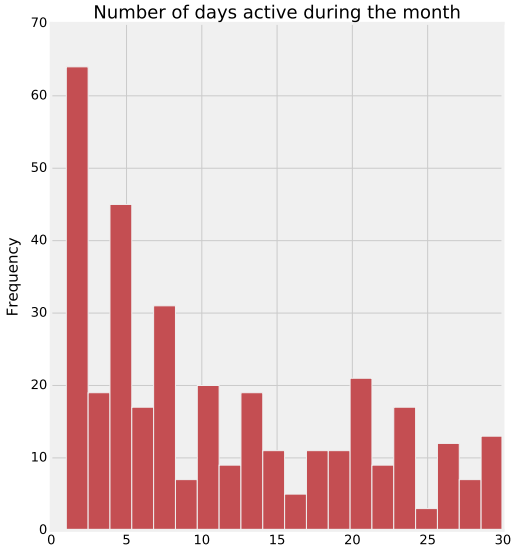
User preferences and biases

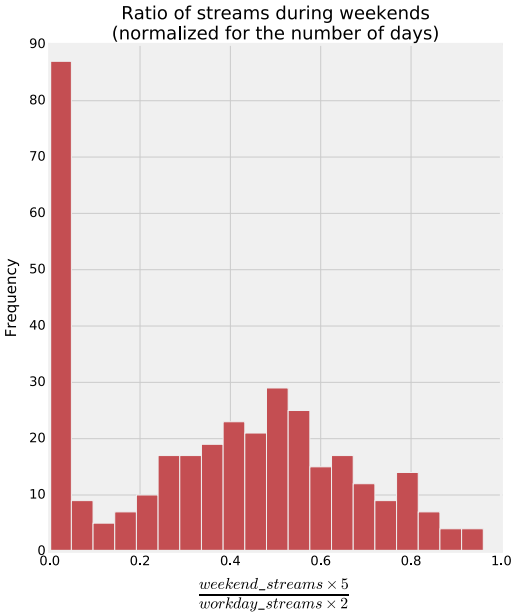
Not all Spotify users are equal. Some are passionate about music and love to discover new artists while others only occasionally stream music and listen to the same songs over and over again.

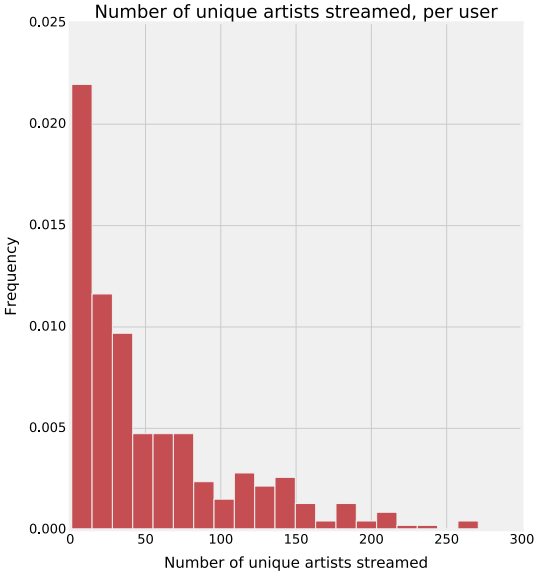
We can verify this intuition by controlling for the other parameters and selecting playbacks sharing the same properties (mobile, premium, and in a playlist created by the user) and see how some aspects of user behavior vary.

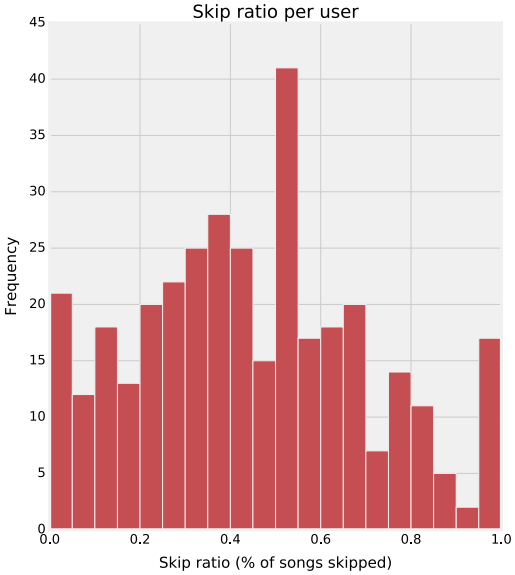












A set of graphs illustrating the variance in user behavior
(from a certain subset of streams: streams from a user's own playlist, with a premium plan and a mobile client)

We kept constant what seems to be the most important factors when it comes to streaming behavior: type of platform used, type of feature used and type of plan subscribed to. In this context, there is no user interface or feature differences between users. However, we still observe a large variance among users on certain aspects of streaming behavior, and a significantly lower variance on others.

The average session length varies wildly for every user, and the same can be said about the number of days spent every month on the platform or the skip ratio (the percentage of songs skipped by the user). All these properties follow a quasi-Gaussian distribution, and exhibit a large inter-user variance, which makes them good candidates as features for predictive or regressive machine learning models on a user level.

Other properties, especially those related to counts such as the number of unique artists streamed, number of days active or number of monthly streams follow a power law. This means that for our particular subset, a large number of users shares similar values, and other values are more rare. Such properties are not as valuable as features for statistical models, but they show that not all properties have a high intra-user variance, especially when controlling for other parameters.

Our conclusion is that taking into consideration each user's properties is not only useful to remove certain biases (such as normalizing by the total number of streams of a user, etc.) but also to build stronger predictive models when using machine learning techniques.

4.2 Modeling streaming context

In this section, we build on our analysis of the impact of streaming context to propose a series of models and methods adapted to music streaming services. The goal of such models can be either to interpret hidden structures in the data that tell us more about users of streaming services, or evaluate the importance and polarity of certain stream events and build a model for user sentiment.

Motivation

As seen in our analysis of contextual information, several features have a high impact on user behavior and have to be taken into account when analyzing music streaming data, or we take the risk of having biased results and taking the wrong conclusions from the analyses we execute. This wealth of contextual information is far from being a curse and comes with a number of advantage, among which a greater potential for predictive models for which every variation in user context can be an additional feature that carries vital information.

There are many ways to deal with such contextual information. One of the commonly used methods is to divide users into "cohorts": groups that are homogeneous with respect to a feature or a set of features. For instance, when analyzing the conversion rate of users, users can be grouped according to the number of months spent on the service and some demographic variables (age, gender, location). The data is then separately analyzed in groups where the users share the exact same values in all these variables.

There are several drawbacks to this method:

- Less data: The number of users matching a cohort's criteria decreases drastically with every control variable introduced. Analyses based on this data are consequently less reliable and more sensitive to noise. It is also a handicap when building more sophisticated models as less datapoints means a higher chance of overfitting.

- Dataset imbalance: A great imbalance can be observed in the size of different cohorts: certain cohorts might be too big and still contain biases from different variables while others are too small and useless from an analytical point of view.
- Scalability: splitting the data in such cohorts is hard to scale, especially given the variety and size of the data used by Spotify.
- Missing on feature interactions: fixing the values of certain variables means that complex effects from the values. they take is hard to observe without a cross-cohort manual analysis.
- Choice of the variables to segment on: The variables to segment on (account age, demographic information, etc.) are often chosen based on some intuition or some superficial analysis of the data, instead of actually looking at the effect of every variable on the targeted metric.

We propose the use of a machine learning approach instead, using contextual information as input features to extract the impact of contextual features on the overall behavior of users, instead of trying to negate this impact.

This method avoids the disadvantages of the cohort analysis approach:

- The data is not segmented and all datapoints are fed to the model (either for training or for validation and testing). This allows us to use models that can be more complex while remaining robust and avoiding overfitting.
- This method is scalable as a large number of machine learning algorithms have highly parallelizable implementations that scale well with the size of the data.
- As long as we choose our models wisely, we can pick up on correlations in the impact of all our variables since no feature is fixed, which can lead to new insights and better performing models.
- At the very least, fitting relatively interpretable machine learning models (such as logistic regression or gradient boosting trees) to the data allows us to find the variables that have the biggest impact on predicting the target, which might be used to choose which variables to segment on for cohort analysis. This can also be used to determine which features to use for taking representative samples when performing A/B testing.

One interesting topic to analyze using this method is skipping behavior. This application allows us to show how a machine learning approach can provide valuable information that takes into consideration all the previously introduced contextual information without the need to segment user, and provide more information than the traditional cohort analysis. But more importantly, skipping songs is a strong but ambiguous signal, and being able to analyze skipped and finished streams with contextual awareness allows us to extract valuable insights about users' satisfaction levels and patterns in their behavior.

Features and methodology

We use the features discussed in the previous sections:

- Platform used
- Feature used
- Premium or free plan
- Reason the stream was started

In addition to these features, we extract user-level features and merge them to every stream done by these user. This is to take into account all user biases discussed in the previous section.

- Session-related features (mean and standard deviation on the session length and number of playbacks, ...)
- Playback-related features (skip ratio, unique artists streamed, ...)
- Time-related features (number of days active, ratio of streams during the weekend over streams during workdays, ...)

These user-level features are extracted from the first week of November, which is excluded from the rest of the analysis to guarantee that we are not leaking the target variable (skipping a stream) through some user features and place ourselves in a realistic scenario.

We pre-process the data by normalizing all numerical feature to extract the normal score:

$$z = \frac{X - \mu}{\sigma}$$

We also transform categorical variables into a set of binary variables (dummy coding). Hence the following table...

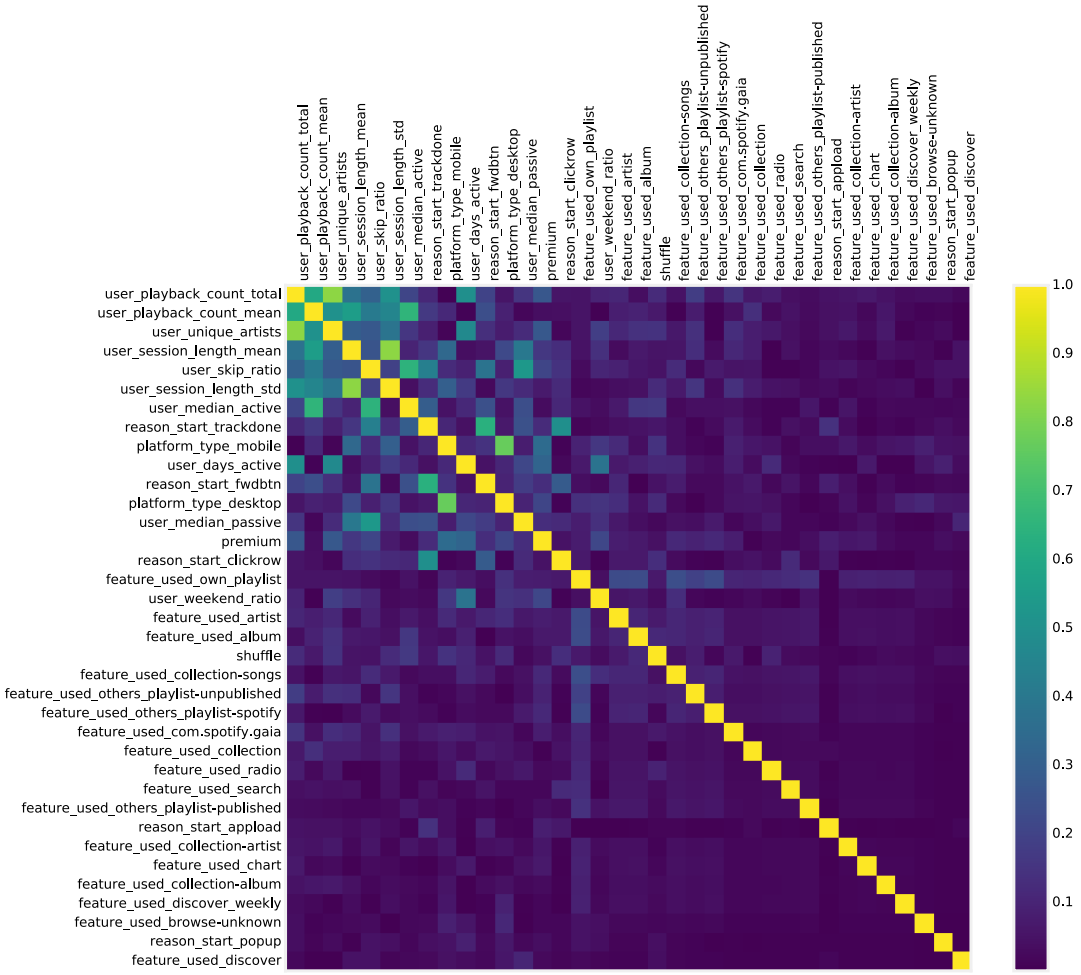
cat_feature
a
b
a
c

...will be transformed to:

cat_feature_a	cat_feature_b	cat_feature_c
1	0	0
0	1	0
1	0	0
0	0	1

To avoid a complete collinearity between these features, we remove one of the resultant features (as it being non-nil is equivalent to all the other features being equal to zero).

Many features (especially aggregated statistics on the user level) are correlated with a large set of features, which means that we should use models that are not too sensitive to this collinearity and use methods that minimize its negative effects on predictive models (by using l2 regularization, projecting to an orthogonal vector space with PCA, etc.)



Absolute feature correlation matrix for a subset of the features

Machine learning models

Since we are looking at a classification task (skipped versus completed streams), many models come to mind. We chose to try simpler, more interpretable models, with a mix of linear and non-linear models: Logistic Regression, Gaussian Naive Bayes, Decision Tree, Random Forest and Gradient Boosting Trees.

Logistic Regression [7] is a generalized linear model generally used for classification tasks. Since in classification tasks we want to find the probability that a datapoint belongs to a given class, this model relies on the logistic function, a function that is in the range $[0, 1]$. With G a linear combination of the input features similar to the result of a linear regression task, we model the evaluated probability as follows:

$$p = P(C = 1|X) = \frac{1}{1 + e^{-G(X)}}$$

By using the logit function (the inverse function of the logistic function), we can revert the problem to a simple linear regression problem:

$$logit(p) = \log\left(\frac{p}{1-p}\right) = G(X) = wX$$

With L2 regularization, this boils down to optimizing the following function, where C is a regularization constant:

$$cost(w) = \frac{1}{2}w^T w + C\sum_{i=1}^{n}\log(\exp(-y_i(X_i^T w + c)) + 1)$$

Gaussian Naive Bayes is a probabilistic classifier, member of the Naive Bayes family of classifiers that are based on applying Bayes' theorem:

$$p(C = C_k|x) = \frac{p(C = C_k)\, p(x|C = C_k)}{p(x)}$$

This family of models has the (naive) assumption that all features are independent. This assumption is rarely true in real-world data, but Gaussian Naive Bayes is still commonly used in information retrieval and text classification where it gives surprisingly good results [32]. This model is also commonly used as a baseline since it is easily interpretable, scales linearly with the number of features and can be solved in a closed form.

Decision Trees are a tree-based decision tool, where a condition is tested at every branch until a leaf node, containing a numerical value or a class, is reached. Such decision trees were traditionally built manually, based on domain language, as a way to automate business or industrial processes. Classification and Regression Trees (CART) [4] is one of the machine learning methods based on decision trees. Instead of building the trees manually, data is used to find the branching conditions that separates the data optimally. What "optimally" means in this context varies from one method to the other, but in general the goal is to increase the homogeneity of the data as much as possible at every branch. In CART, the metric optimized is Gini impurity, that can be defined as follows (with f_i the portion of items of class i):

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i=1}^{m}(f_i - {f_i}^2) = \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} {f_i}^2 = 1 - \sum_{i=1}^{m} {f_i}^2 = \sum_{i\neq k} f_i f_k$$

Random Forest [19] is an ensemble learning method that works by training a given number of trees and evaluating every tree on the data we want to classify and calculating the mode of all predictions in the classification case, or the mean in the regression case. Random Forest have given great results in many fields, including finance, biosciences and health-care. Other than the decision tree learning algorithm described above and the general advantages of ensemble methods, what really explains why Random Forest performs well on many problems is the way each decision tree is trained: at every branch of the tree learning algorithm, a random subset of features is chosen, and bootstrap aggregating (bagging) [3] is used to generate a new training set. Choosing a random subset of the features reduces the correlation between multiple trees, as features that strongly predict the target would otherwise be present in most trees, negating the benefits of ensembling different predictors. Bootstrap aggregating reduces the variance of the model without a significant increase in bias.

Gradient Boosting Trees [9] is a special case of gradient boosting using decision trees. This machine learning technique builds a prediction model (for regression or classification) by combining an ensemble of weak predictive models. The output of this model is a weighted sum of functions $f_i(x)$ from some class of weak learners (in the case of gradient boosting trees, this class is classification and regression decision trees).

$$F(x) = \sum_{i=1}^{M} \gamma_i f_i(x) + \text{const.}$$

The weak learners are iteratively fitted to pseudo-residuals, minimizing the average value of an arbitrary loss function using the principle of steeped descent.

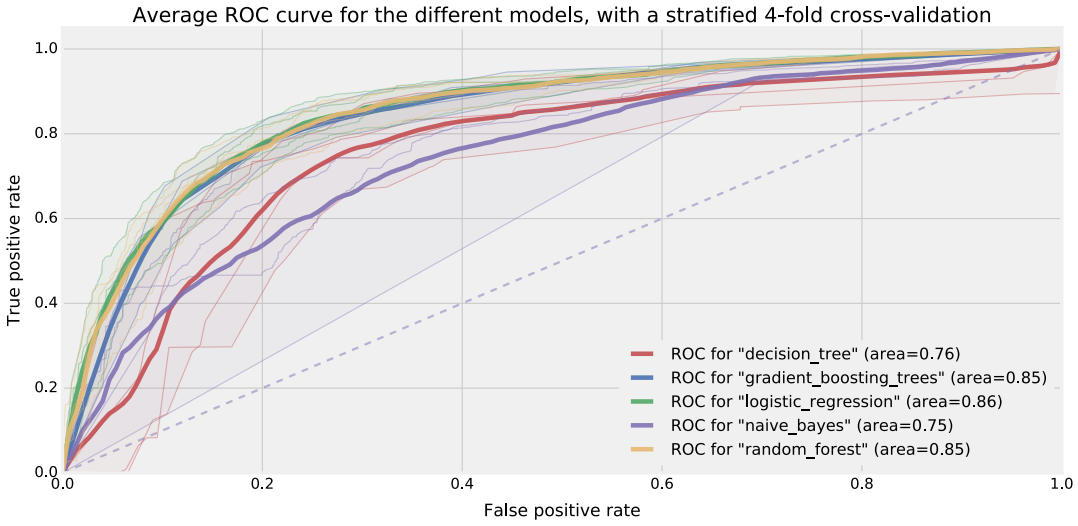
The optimal parameters for these models were found by doing a Randomized Parameter Optimization [2] over the parameter space and using the area under the ROC (Receiver Operating Characteristic) curve as a target to optimize. Here follows are the optimal parameters for these models:

- Logistic Regression
  - l2-regularization, with an optimal regularization strength found by cross-validation
- Gaussian Naive Bayes
  - No hyper-parameters: The means and variances are estimated using maximum likelihood
- Decision Tree
  - A maximum tree depth of 5
- Random Forest
  - A maximum tree depth of 4
  - A minimum weighted fraction of 15%
  - 80 trained trees
- Gradient Boosting Trees
  - A learning rate of 0.01
  - A maximum depth of 2
  - 45 trained trees

Model evaluation

We evaluate these models by splitting the data from the last three weeks using a stratified k-fold cross-validation (for `k=4`) and calculating the area under the ROC curve (ROC AUC), a method that is standard for evaluating machine learning models as it allows the comparison of multiple models and shows which models give the best balance between false and true positives.

Here follows the ROC curve for the evaluated models:



Decision Tree and Gaussian Naive Bayes predictibly lag behind: Naive Bayes is not suited to deal with tasks that include highly correlated variables (because of the independence assumption that comes with this model). If both models seem to have a rather decent decent performance in average, this wildly varies from one cross-validation batch to the other, as both models are known for being sensitive to overfitting to the training data (even with regularization constraints) and being sensitive to slight changes in the distribution of the test or training test. A model that overfits the data is not appropriate since our main goal is to build an interpretable model that generalizes well.

The other results are however rather unexpected: wildly different models (Logistic Regression, Decision Tree and Gradient Boosting Trees) have a similar performance, with only slight variations, as if there was a ceiling that could not be surpassed.

Although surprising at first, this result is coherent with our task: we are trying to predict skipped songs, which is a decision that does not only depend on the contextual features alone, but also on the song being currently played and the user's mood and a multitude of other factors. In other words: given the exact some context, a user might skip or complete a song, and our goal is simply to pick the most likely candidate.

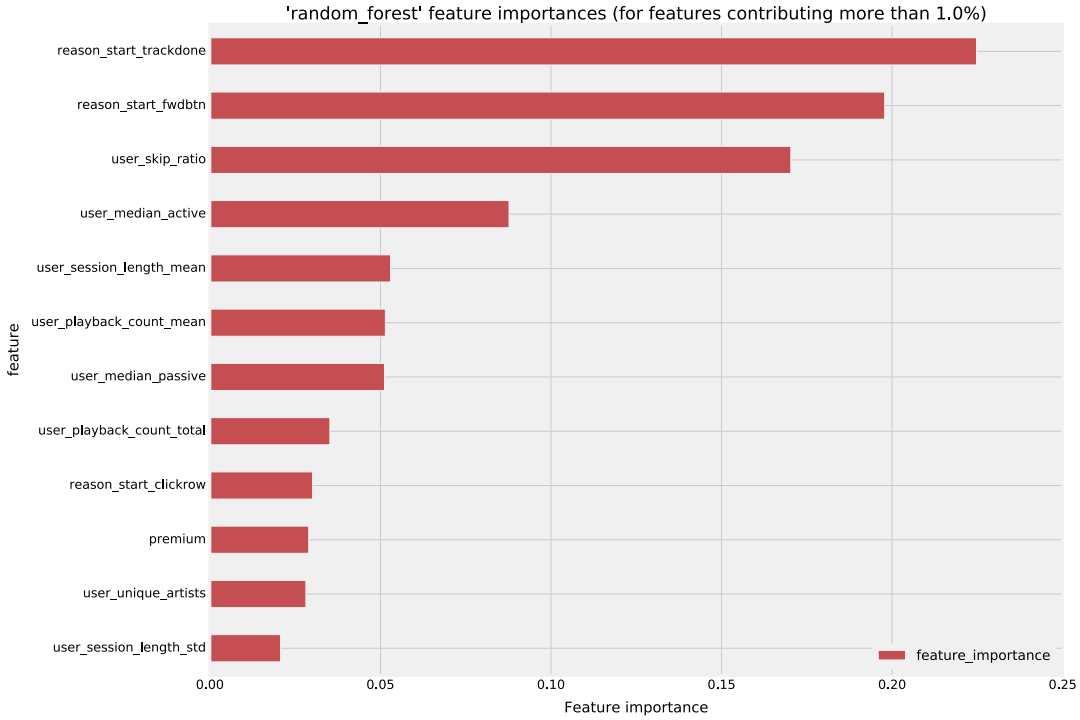
It is still notable that we reach such a high level of accuracy with contextual information alone: this reinforces our believe that contextual information is vital when it comes to analyzing music streaming behavior.

Given that three models have an equally good performance on this task, we might be tempted to follow Occam's razor and choose the simplest viable model: Logistic Regression. However, this linear model does not leverage arbitrary feature interactions (such as the interaction between platform type and type of account) whereas non-liner model such as Gradient Boosting Trees and Random Forest learn these relationships. This means that such models will have a better representation of the structure of the data and the features that impact skipping the most.

We choose to use Random Forest for a more exploratory analysis, but other models can be interested for other use cases (such as the one shown in section 4.3)

Feature importances

We train Random Forest with the previously used parameters on the entirety of the data and analyze the relative importance of every feature used by the model. The graph below shows the top ranking features and their respective importances:



Among the top ranking features, we note that the most important features by far are those related to the action starting the stream. This shows that there is a strong link between successive events (such as skipping and finishing a song), which we showed in the previous section and we will develop in more details in our study of the sequential analysis of streaming events in chapter 6.

We also notice that a large number of variables, describing different facets of streaming context, have a significant importance: most important of which (and predictably so) is the `user_skip_ratio` and other variables calculated on a user level, `feature_used_radio` and other variables which relate to the feature used for streaming, `platform_type_mobile` and other platform indicator and finally the type of plan used by the user (`premium`) which plays an important role in the model.

Like said in the introduction of this section, we can use these feature importances to determine which features to isolate for certain types of analysis (like A/B testing). For example, when analyzing the effect a feature has on the skip ratio, we should take into consideration the previous skip ratio of the users in the control group (or try to sample users for these groups to preserve the same distribution of skip ratio observed among all users).

These insights are valuable to interpret the totality of the data, but it can also be ran on a subset of users to learn its specificities. However, such models can be used even further and provide per-stream metrics, such as uncovering the ambiguity behind different reasons a user might skip a song.

4.3 Stream polarity and user sentiment

In this section, we leverage the model previously built to infer positive or negative sentiment in every stream using only contextual information and the actions taken by users.

Motivation

Like previously said, playing or skipping songs is the strongest signal at our disposal: the intuition is that users like songs they finish, and dislike songs they skip. There is however more nuance to it, and skipping can be ambiguous and does not necessarily induce a strong disapproval of a song (as we saw in the previous chapters analyzing the heavy impact of multiple factors on skipping). Because of this, only positive feedback is usually extracted from streaming services, and the number of times a song was played is used as a confidence measure. Not having negative feedback is one of the biggest weaknesses of recommender systems based on implicit feedback [16].

In this section, we propose a model for stream polarity, based on the previously introduced contextual model. We show different options to leverage that model to calculate a stream polarity, and show some initial applications for this polarity, such as inferring user sentiment towards the service.

Leveraging skipping likelihood

The basic intuition behind our model is that the more unexpected an event is, the more important it is.

If the user is in a context where they are very likely to skip (for example: right after skipping a previous song, on mobile, with a premium account), skipping is not as negative (because regardless of the user's sentiment towards the song, the context encourages it) but finishing the song is a more positive signal (because even in a context biased towards skipping. the user still chose to finish the song).

In other words, we are interested in he deviation between the likelihood of a user skipping and their actual behavior.

There are many ways to take this into consideration, the most simple of which being the following linear relationship:

$$stream\_polarity = P(skip) - skipped$$

$skipped$ here represents the user's actual behavior, and is equal to 1 or 0 respectively if the user skips or finished the song.

This allows to have a polarity between -1 and +1, where unlikely skips are closer to -1, unlikely finished songs are closer to +1 and likely events are closer to 0.

This idea of giving bigger weights to predictions that have both wrong and have a high confidence reminds us of the concept of "log loss", which is generally used as a loss function penalizing such errors and minimized when training classification models such as logistic regression.

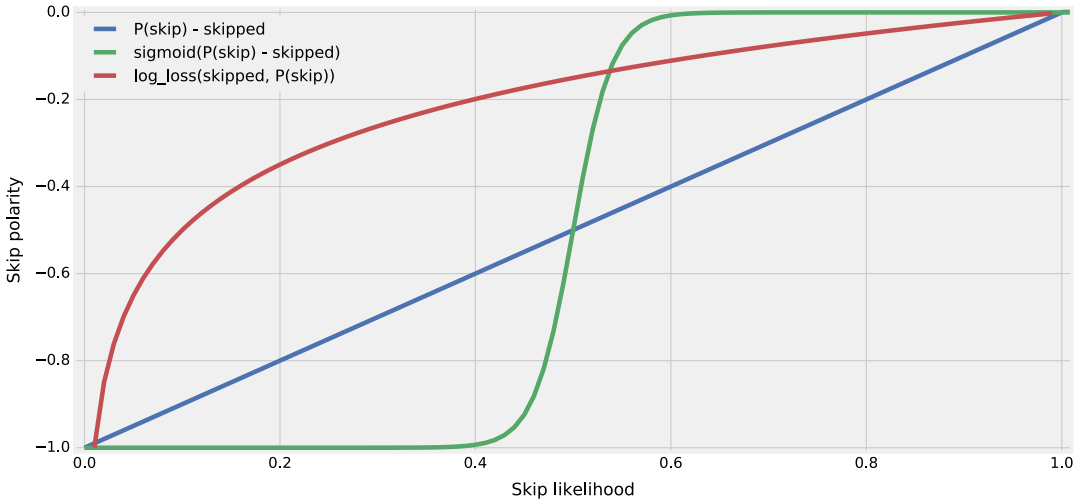
Log loss for a single prediction is defined as follows:

$$y_i \log(P_i) + (1 - y_i) \log(1 - P_i)$$

Where P_i is the likelihood of the datapoint being positive, and y_i is either 1 or 0, depending on whether the datapoint is positive or negative. Log loss is not for $P_i \in \{0, 1\}$, but it can be clipped for a chosen epsilon and normalized to be in the interval $[0, 1]$.

Another possibility, when trying to have a more contrasted output, is to simply apply a sigmoid function on the resulting "unlikelihood". This is commonly used in neural networks to get a binary output from continuous outputs.

The following graphs show how stream polarity for a song that was skipped would change with the skipping likelihood. Note that log loss is clipped and normalized to fit in the desired range.



The sigmoid polarity would be useful for binarizing skips, if our goal is for instance to radically weight-out likely skips or finished songs to only take into account the unlikely ones.

Log loss heavily penalizes wrong predictions that have a very high likelihood. This would result in a tighter distribution of polarity values, with polarity drastically decreasing as the likelihood gets closer to the actual result.

For inferring user sentiment, log loss and the linear polarity seem like the best choices. In fact, our linear polarity is the same as log loss if we do not apply the log function to the skip likelihood, so in a sense the results should sensibly be the same, and we would need to further evaluate these two choices in each application to know which works better.

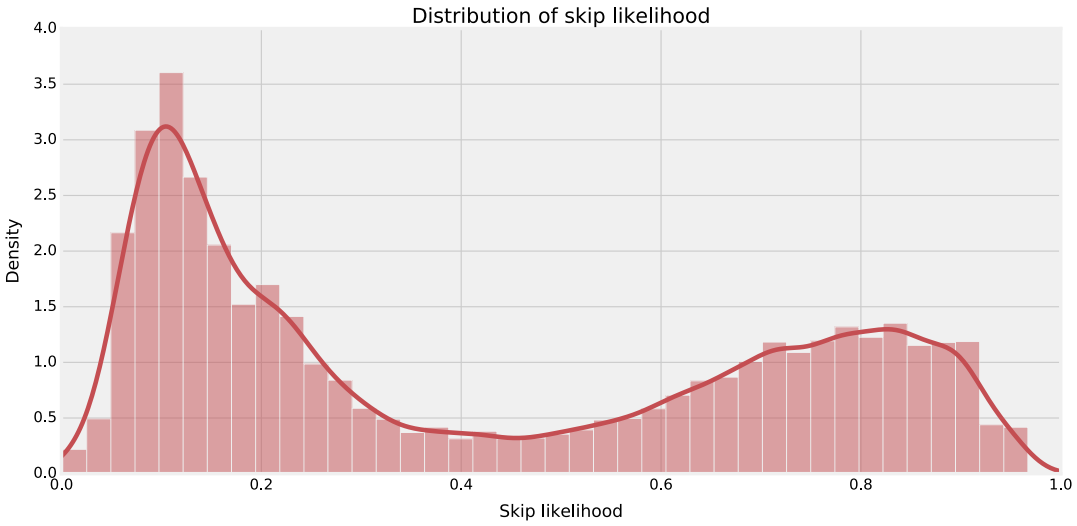
For the needs of the study, we choose the simpler linear polarity as it is easier to interpret and keeps all the variance produced by the model.

Calculating skip likelihood

We previously used a Random Forest to analyze the influence of various features on skipping behavior. We also noted that other non-linear models such as Gradient Boosting Trees are appropriate to analyze the structure of the data. However, when it comes to using the classification likelihood, these models might not be adapted: they learn a sparser set of variables than logistic regression (with l2 regularization) and by design do not provide an easily interpretable likelihood estimates. These estimates have very little variance in the case of Random Forest, because of its reliance on the clear-cut decision functions learned by shallow trees, whereas Logistic Regression gives skipping likelihood estimates that have more variance, covering the whole spectrum of possibilities and taking into account every contextual variable.

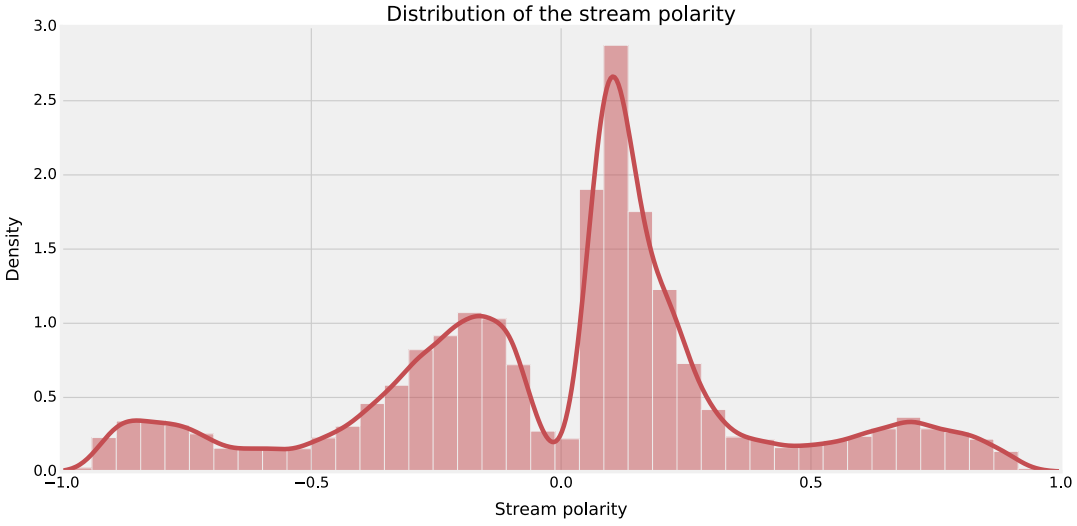
For these reasons, we chose to use Logistic Regression as a way to evaluate the skip likelihood for every stream.

Similarly to the previous section, we extract user properties from the first month of November and fit the model on contextual information from the remaining three weeks in addition to the previously calculated user properties.



The first observation we can make from looking at this distribution is that its variance is really high, with streams being all over the spectrum. This shows once again the possible variety of contexts where songs are streamed, and why segmenting with respect to all parameters is not feasible.

Unsurprisingly, the likelihood distribution is bi-modal with one class of streams having a high likelihood of being skipped while the others have a significantly lower likelihood. This is reflected in the distribution of stream polarity where both positive and negative streams have two modes:

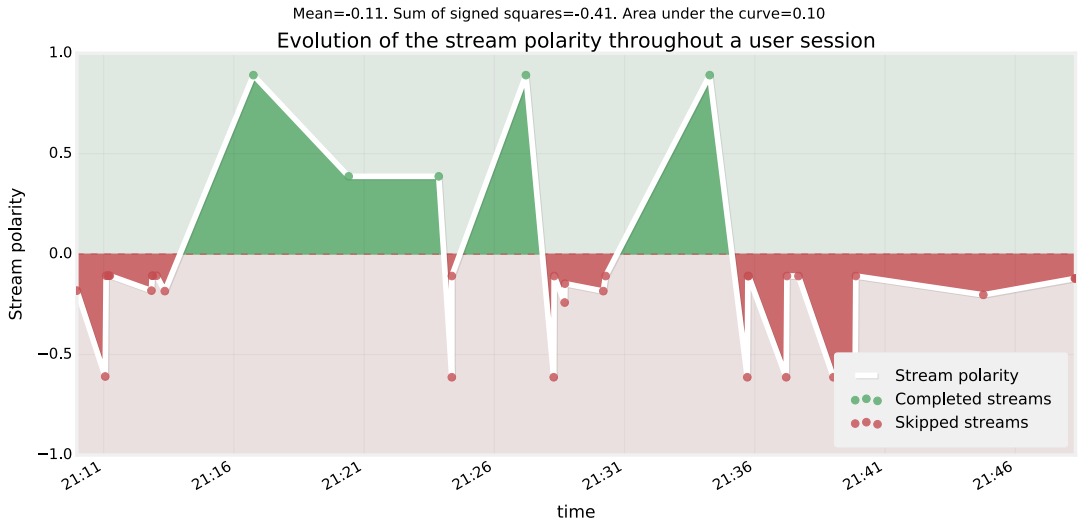


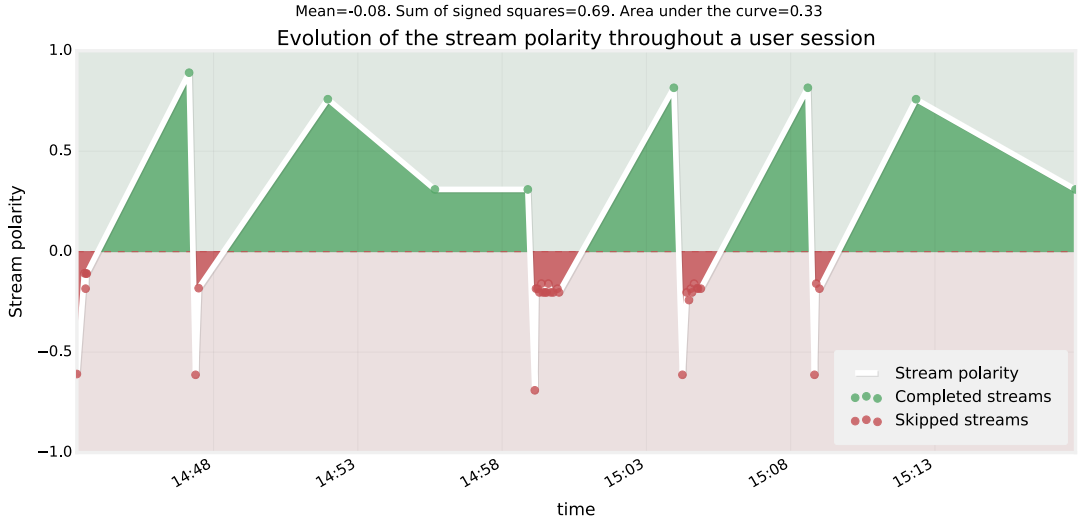
We notice that most streams have a positive polarity, which can be explained by the fact that more songs are finished than songs skipped, but the proportion of streams that have a low absolute polarity is higher for these positive streams. This is because users, among all platforms and contexts, commonly play a large succession of songs only rarely skipping. This can also be a proof of "survivor bias": over time, users that are dissatisfied with the service will leave it, and the distribution of sentiment for users still in the service is biased towards a more positive sentiment.

Even if these general insights are interesting, the real power of stream polarity appears when we aggregate it to infer user sentiment in a session, their preference towards some certain songs and artists and their satisfaction with certain features or the service as a whole.

Session polarity

Simply looking at the evolution of this streaming polarity throughout a user session gives us an idea on how satisfactory the session was, and how polarity changes with the user's actions:





The first session looks like it was not a very satisfactory one for the user: only 5 songs were completed, and most of the session duration was spent skipping and selecting songs. We can notice that the negative polarity of these events varies (as the user switches from one feature to the other, or skips the songs in different ways).

The second session looks more satisfactory, with 8 completed songs, and a shorter period spent skipping. We notice that skips are focused in "clumps" of rapid successive events. Because we take the last action into consideration, only the first skip (which made the user tran-

sition from listening to a song to this impulsive skipping behavior) has a highly negative polarity, whereas the others have a polarity that is much closer to being nil. This is a positive because this behavior is common among users (that skip through a large number of songs in a short period simply to reach a song they are looking for). We will discuss the arrival of this skipping events more in details in the two following chapters.

There are many ways to aggregate this polarity on the stream level:

The simplest approach is to simply calculate the mean stream polarity over all streams in the session. This works well for simple sessions where there is a small number of highly polarized streams, but most sessions are more complex, with a large number of streams with a low polarity that will bias this mean towards 0.

$$mean\_polarity_{session} = \frac{1}{\|session\|} \times \sum_{stream \in session} polarity_{stream}$$

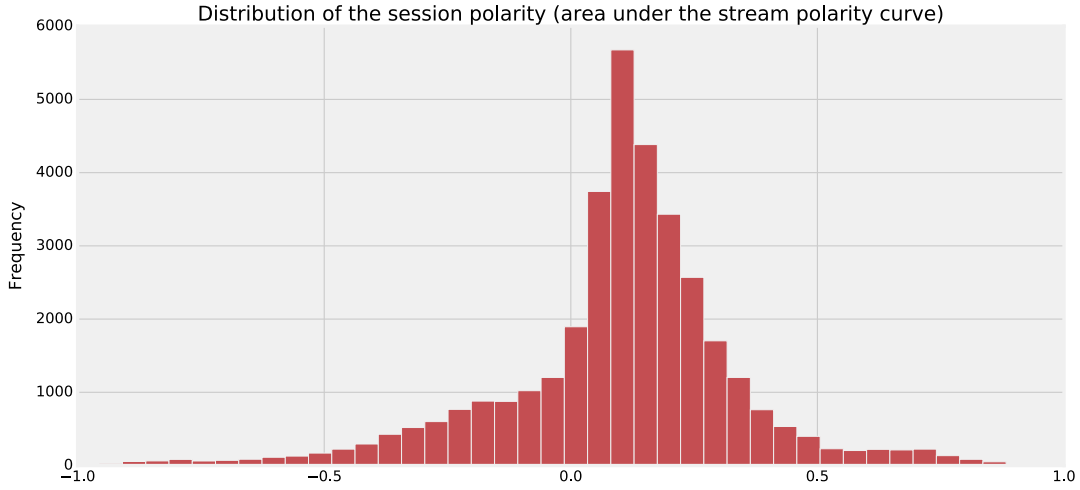
Another approach to counter this problem is to emphasize the effect of very polarized streams (as we previously seen with the use of log loss or sigmoid). One way to do so is to calculate the sum of squares of all streams' polarity while taking their sign into consideration. This gives us a better idea in most case, by negating very unpolarized streams (such as multiple skips arriving one after the other). However, normalizing this value by the number of streams leads to the same issue, and the lack of normalization makes it hard to use this metric to do intra-session comparisons. Most importantly, this still does not take into consideration the chronological aspect, since a series of events arriving in a short period will still have a bigger impact than events having a long duration, whereas it seems intuitive that a session where user is in a negative state for a longer period of time should be more negative than one where he is in a negative state during a shorter period of time.

$$squared\_polarity_{session} = \sum_{stream \in session} sign(polarity_{stream}) \times polarity_{stream}^2$$

The method we propose combines both the stream amplitude and the chronological aspect: we use the area under the polarity curve as a measure for the polarity of sessions. This still uses the information given by every stream's polarity (since a succession of highly polarized streams means the surface will be higher) but also integrates the chronological element into the equation. Additionally, this area is easy to normalize by simply dividing by the length of the session. As such we get a metric in the range $[-1, 1]$

$$area\_under\_the\_curve_{session} = \frac{1}{t_f - t_0} \times \int_{t_0}^{t_f} polarity_t$$

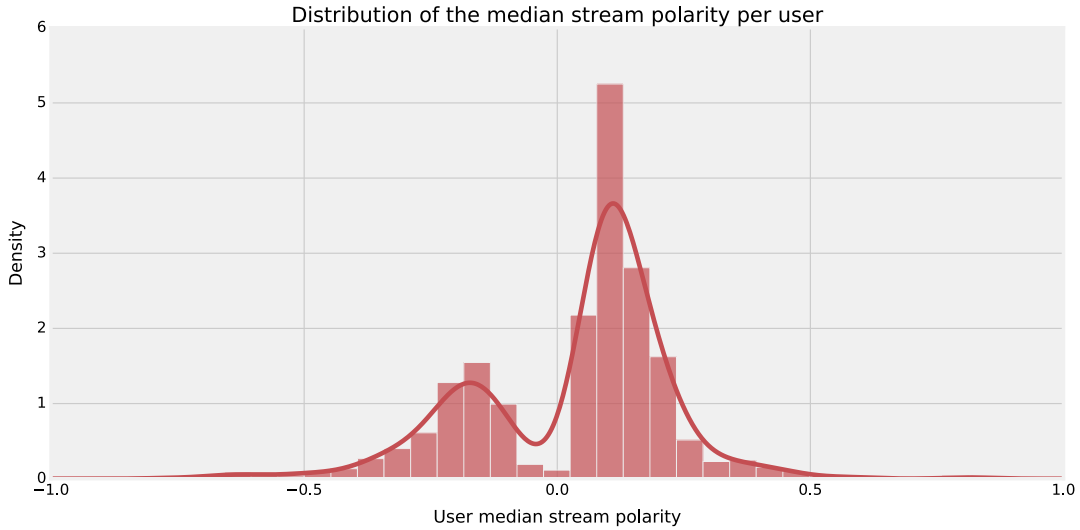
This last metric is biased towards positive values (for the same reasons there are more positive than negative streams), but we can remediate to this by simply deducting the average session polarity if our goal is to compare sessions to the average case.



User sentiment

We can apply a similar logic to users, and aggregate stream polarity among all of these users, either by using the previously calculated session polarity and aggregating over users or directly using every user stream's polarity.

We choose the latter method as this gives us a clearer separation between two classes of users, as is shown in the plot below:



Interestingly enough, this aggregated user polarity follows a bimodal distribution where very few users have a nil polarity, with users distributed over one positive gaussian distribution with a mean of +0.11, and one negative distribution with a mean of -0.17. Since we used the median, users with a positive polarity are those that have more positive events (finishing songs, etc.) than negative songs.

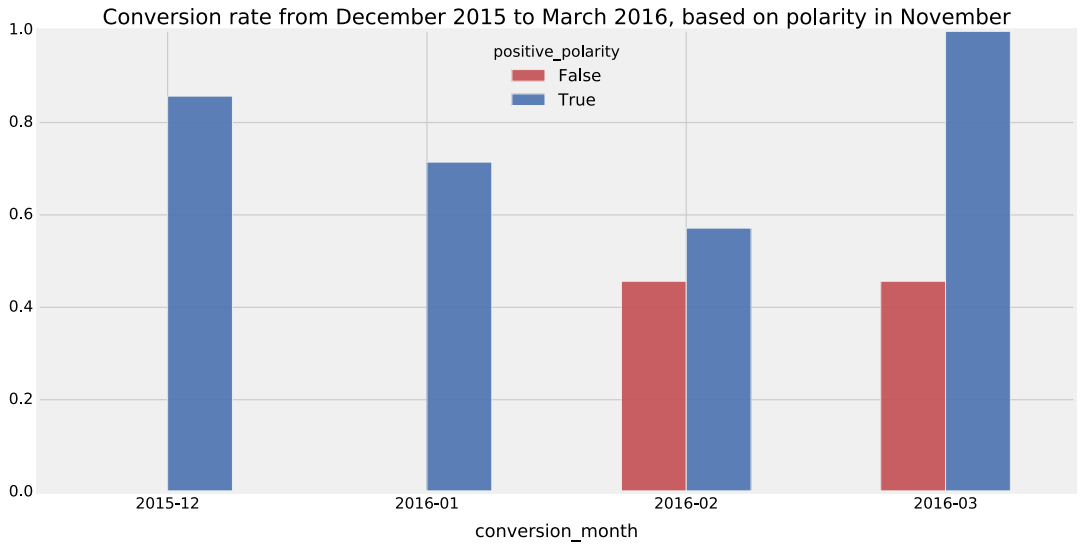
To evaluate the properties of these two classes of users (positive and negative polarity), we extracted data relative to user retention, activation and conversion. These terms are easily confused, but each one has a very specific mean and are commonly used in tech companies,

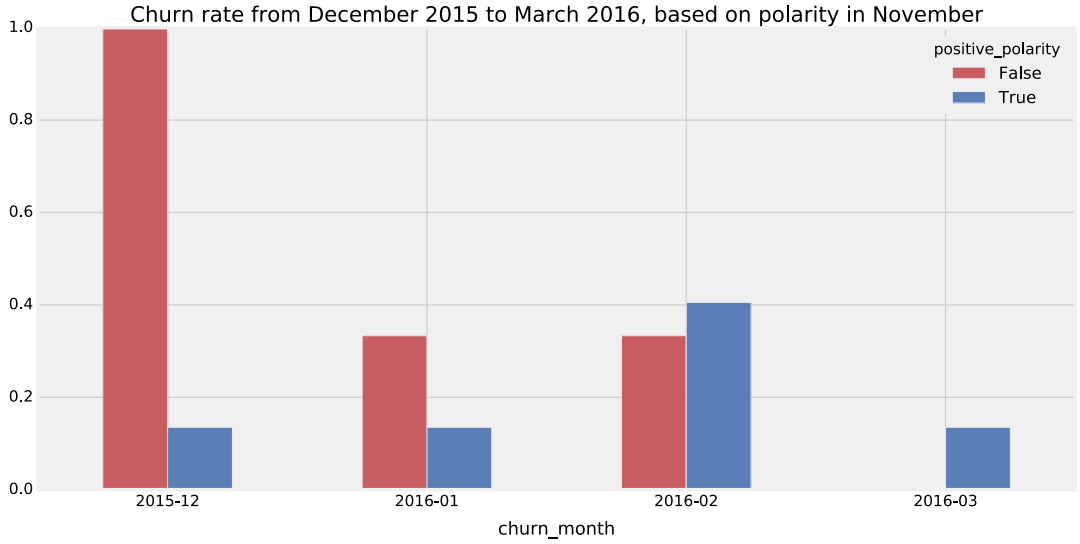
especially those that deal with recurrent users and the duality between free users and subscribers:

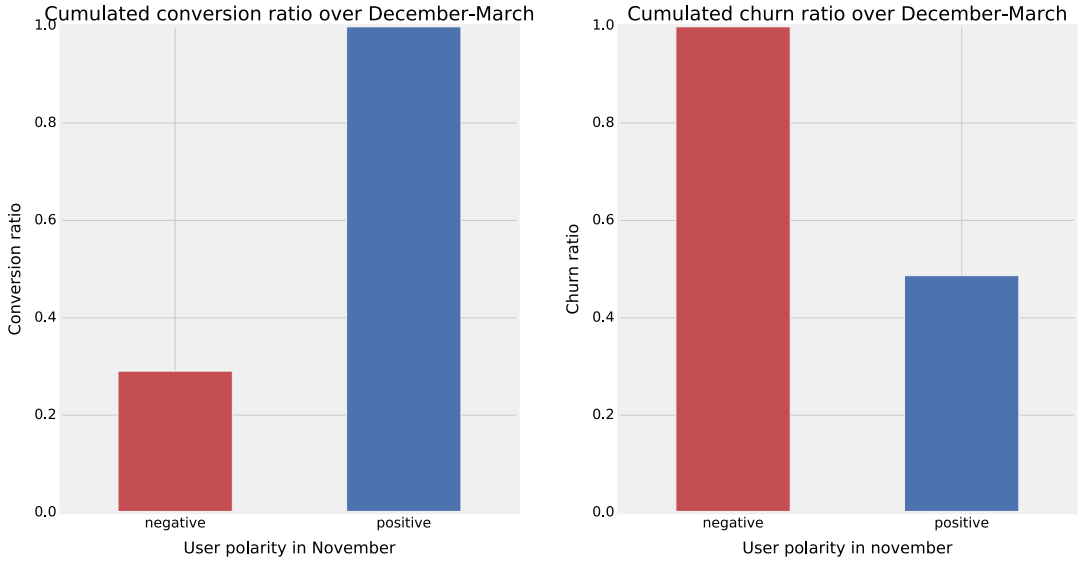
- Activation is the percentage of total users that are effectively using the service. In the case of Spotify, this is someone that streamed a song in the given period.
- Conversion is the percentage of users that were converted, generally users that bought an item, or in the case of Spotify users that went from a free plan to a paying plan.
- Retention is the percentage of paying users that stay on the service. We often focus on the opposite phenomenon, called churn, which is the percentage of users that leave the paying plan.

Our hypothesis is that users of the free plan that had a good experience on the service are more likely to convert to a premium account than users that had a bad experience. To confirm this, we calculate the conversion and churn rates over three months (from the beginning of December 2015 to the end of March 2016) and see if the positive or negative experience users had in November 2015 had any impact on these rates.

The actual conversion and churn rates were normalized to avoid leaking sensitive information. As an indication, these rates (over a short period such as the one analyzed here) are usually relatively low, especially in a service like Spotify.





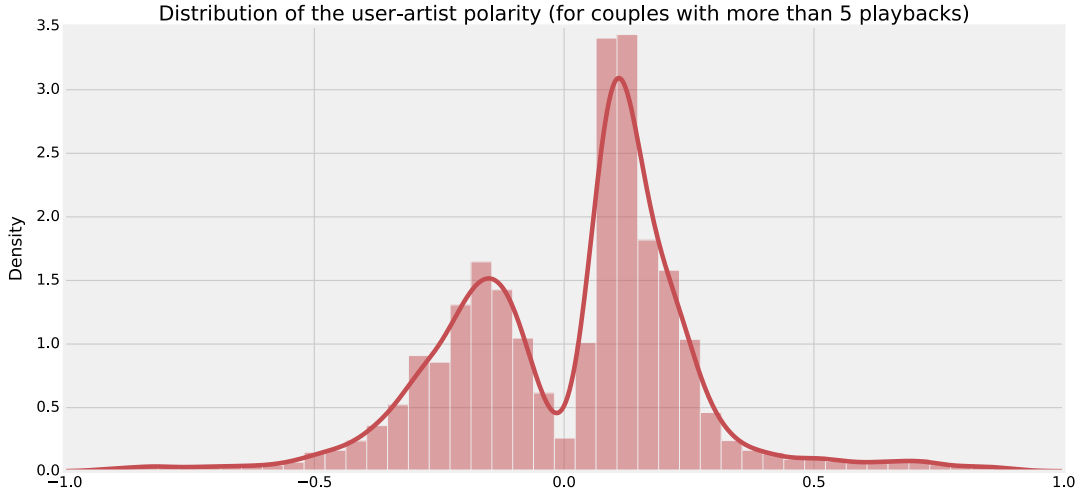


We can observe that not only the aggregated conversion rate is significantly higher for free users that had a positive polarity in November, but they also convert earlier than other users. The inverse remark can be done about churning: premium users that had a positive polarity in November are significantly less likely to churn overall, and if they do they most often do it later.

One caveat to the results above is that the small number of users used in this analysis. Since we are only using a small subset of the data, and only looking 4 months ahead, a small percentage of the premium users churn, and similarly only a small percentage of the users convert to the premium offering. We discuss theses limitations in the section 7.1.

Additionally, we can also do the same type of analysis when it comes to determining the relationship between users and artists, for instance to know which artists users like or dislike. One important thing to take into consideration in that case is that we might need to only keep the most positive interaction the user had with a certain song, as that gives a better idea of the

sentiment towards a song or an artist (even if a user likes a song and completed it in the past, they might want to skip it in certain circumstances).



4.4 Conclusion

We showed examples of how multiple contextual parameters such as the platform used, the type of plan or the reason a stream was started can impact multiple facets of user behavior such as skipping, the variety of songs listened to and the average time spent on the service.

This rich contextual information can be used to analyze user behavior without the impact of certain biases. However, the high number of variables makes this challenging, and traditional methods such as as cohort analysis presents many drawbacks, especially given the interactions between these variables and the volume of contextual information that needs to be analyzed.

To counter these limitations, we proposed a machine learning approach that allows us to analyze the impact of these contextual variables. Since skipping plays such a high role in inferring user behavior and in inferring their preference, we used it as a target for our models and evaluated their performance on this task. By fitting one of the best performing non-linear models, Random Forest, we were able to visualize the impact of every variable on skipping behavior.

Last but not least, we used a similar approach (with a different model: Logistic Regression) to infer the polarity of every stream by the user. This polarity is based on the intuition that the more an event is expected, the more it is important: finishing a song in a context particularly prone to skipping has a higher polarity than doing so in a context where the user would most likely finish any song. We showed how this polarity measure can be used to track and monitor user satisfaction, classify sessions and infer users' preference and sentiment towards artists and songs. These applications can help improve the input given to recommender systems, have a better understanding of user behavior and build additional business metrics to be used for monitoring and for product development (in the context of A/B testing).

One of the limitations of this user sentiment model is that one type of variable has a drastic impact on the polarity: the reason a song was started. This is because user actions are tightly coupled, and active events (such as skipping or clicking on a song) most often arrive in quick successions. We will study all the implications of this observation with a sequential analysis of user actions in chapter 6.

Chapter 5

User attention

In this chapter, we will focus on an often underrated facet of data analysis on user-facing products: attention. We present why attention is a vital metric to monitor in a media consumption service like Spotify, we analyze some simple and readily available metrics to do so and propose a series of relevant models for user attention.

5.1 An attention economy

We live in an attention economy: Every year, thousands of new tech companies are competing for our attention, want to make us hooked to their mobile games or social networks and spend sleepless nights trying to improve the mean time spent on their platforms by 30 seconds.

The idea of information overload and its inevitable consequence, the scarcity of its recipients' attention, has been brought by Herbert A. Simon [29] in the seventies, but it is only later, in "The Attention Economy: Understanding the New Currency of Business" [8], that the term attention economy was coined. In this book, Thomas Davenport and J. Christopher Beck define attention as follows:

> Attention is focused mental engagement on a particular item of information. Items come into our awareness, we attend to a particular item, and then we decide whether to act.

This definition applies particularly well to the case of streaming services: these services can be used in a state of passivity where items are being played in the background without any mental focus from the user, or the user can be mentally focused on a particular item, for example when they realize that they dislike the video being streamed and skip it.

However, user behavior in the case of music stream differs greatly from video streaming, mainly because of the difference in content (length and frequency of consumption) and the fact that listening to music does not require constant attention from users, as noted in [31]. For instance, whereas it is rare that people watch a whole movie while performing other actions and not paying attention to the streamed content, it is rather common to stream music in the background while working out, cooking or taking a shower. It is also common to start playing music from a playlist, and forget to turn it off when leaving one's workstation.

Being able to catch users' attention is not necessarily a positive thing: certain services aim to minimize the interactions needed to function, and hence aim to be necessary while requiring minimal levels of attention. But identifying this type of usage and separating it from a more active interaction between the user and the system is nevertheless interesting.

For all these reasons, it is vital to monitor the fluctuations of users' attention in an automated way, as simple polls or user studies are expensive and hard to deploy at the scale of a service such as Spotify.

In the following sections, we present signals from which we can infer the user's attention, simple metrics that can be calculated from them and more complex models for user attention.

5.2 Analyzing user attention

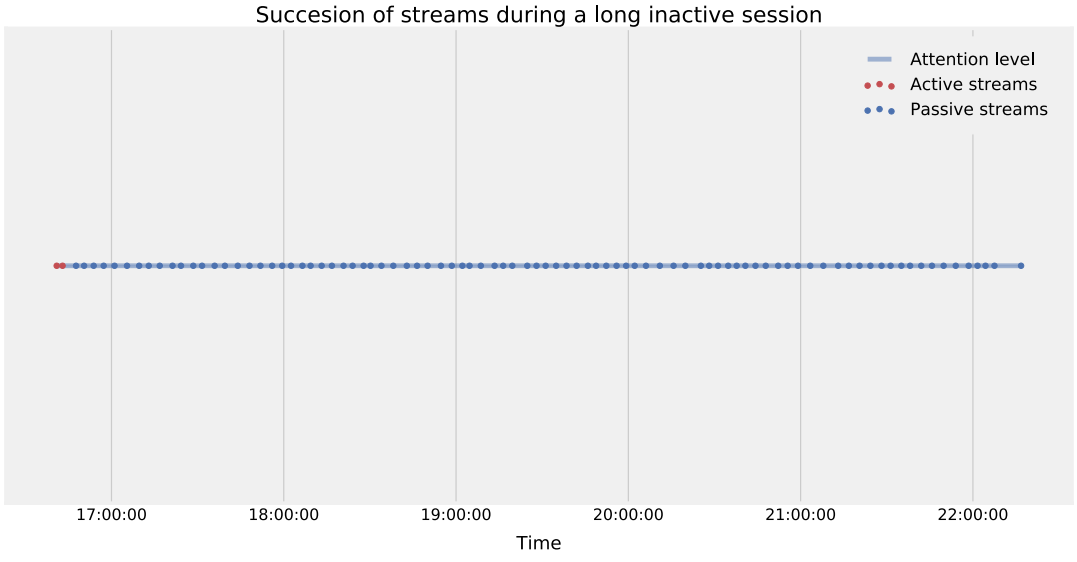
In this section, we will build a simple model for user attention and evaluate whether the statements above are confirmed by the data, and discuss how attention level can be used in the context of modeling users' experience in the case of music streaming.

Sessions and activity level

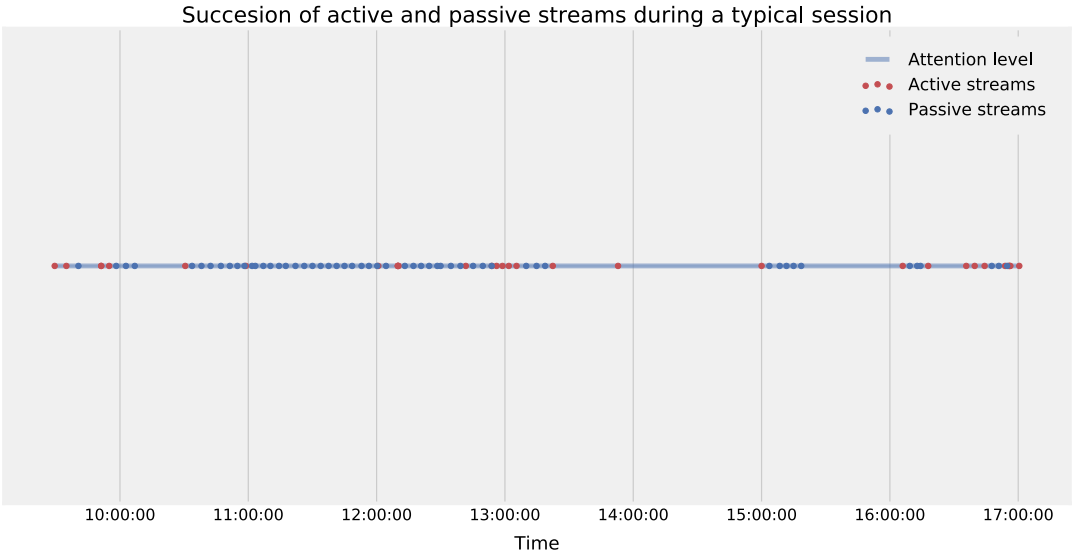
Spotify collects multiple signals from its clients, including which actions users are performing on the interface, which pages they are viewing and the reason why a track was ended or started.

This information can be used to segment streams into user sessions, where a session consists of a series of events (tracks played and/or pages viewed) that are separated by a duration lower than a certain session threshold (around half an hour of inactivity). Although this already models a certain notion of activity vs passivity, it does not take into account that even in the absence of the user, tracks might continue to play through a playlist, or one of Spotify features which continuously plays tracks. It is hence common to find extremely long user sessions (up to 4 or 5 hours long sometimes) as a result of the succession of playbacks where users are in fact inactive, such as music playing in the background while performing sports or having dinner, or a user that simply forgot to close the client.

We can analyze the nature and activity level of these sessions by separating two types of streams: active streams that imply some action from the user (skipping a song, seeking through a song, picking a song from a playlist, ...) and passive streams (songs that were completed without a user intervention)



The plot above shows the example of a session that lasted more than five and a half hours, whereas the average session length in our sample is significantly less than an hour. Such sessions are most often constituted of a large succession of playbacks (90 in the session above) of which only one or two were triggered by the user.



Unlike the previous session, these two sessions (collected from the author's listening history on the 4th of November) exhibit a type of behavior common among many users: A quick succession of multiple active streams (songs being skipped or selected from a playlist) followed by long uninterrupted sequences of streams. This behavior is fairly common on the service, and shows the limit of a frequency based approach:

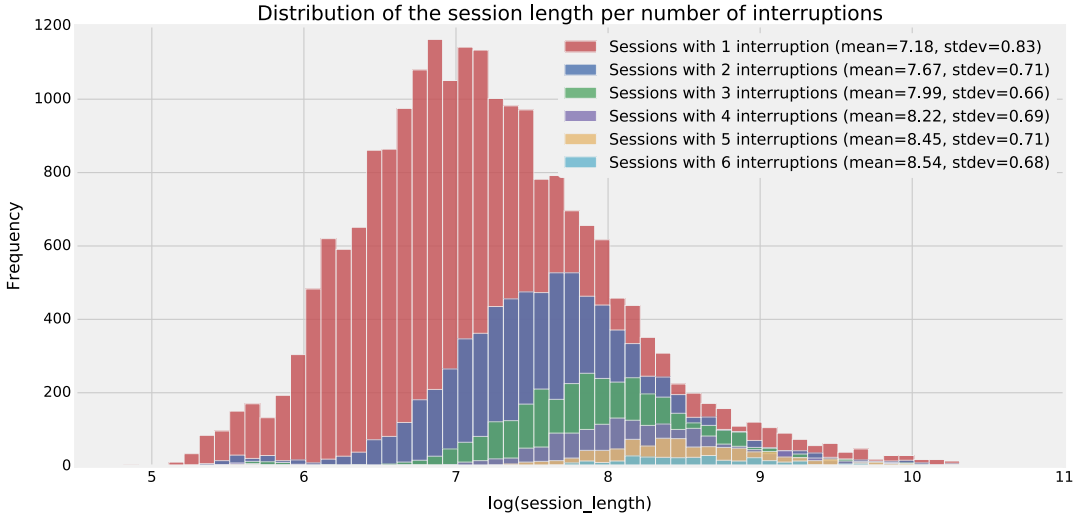
- The arrival of active events (such as skips) does not follow a constant distribution, and is influenced by type of streams that preceded it.
- Skips and other active events usually happen one after the other (when a user is skipping through a playlist, for example). Simply calculating the skip ratio considers every one of these successive streams as equal, even if the first one is the most meaningful (as it interrupted a song, whereas the others are simply a mechanical way of moving through a playlist or the radio mode for example).

Importance of the attention level

Long passive sessions are not intrinsically bad for the service, or a proof of the user's dissatisfaction, as many users like to play music in the background without interacting with the clients. But it is important to take this into consideration when analyzing individual playbacks as the important of such streams, especially the ones that happened 3 or 5 hours after the start of a session, is uncertain as the likelihood that the user is still listening to these tracks is significantly lower than that of streams that occur right after some user actions (clicks, skips, ...). This is particularly useful when evaluating the effect of new features, whether they promote more active or passive listening, and for recommendation systems where it can be used as a confidence metric that a completed song was actually liked by the user, especially in the absence of multiple playbacks of a song which is usually necessary in the case of implicit feedback [16].

Similarly, sessions that exhibit a higher level of attention from the user, with multiple sequences of streams split by periods of activity, can help us to get a better understanding of users' preferences, as we have a higher confidence that the songs played were appreciated by the users. We can also look at this from a product development perspective: knowing that a feature or a specific type of platform correlates with higher levels of activity is a valuable information when introducing new features or evaluating current ones.

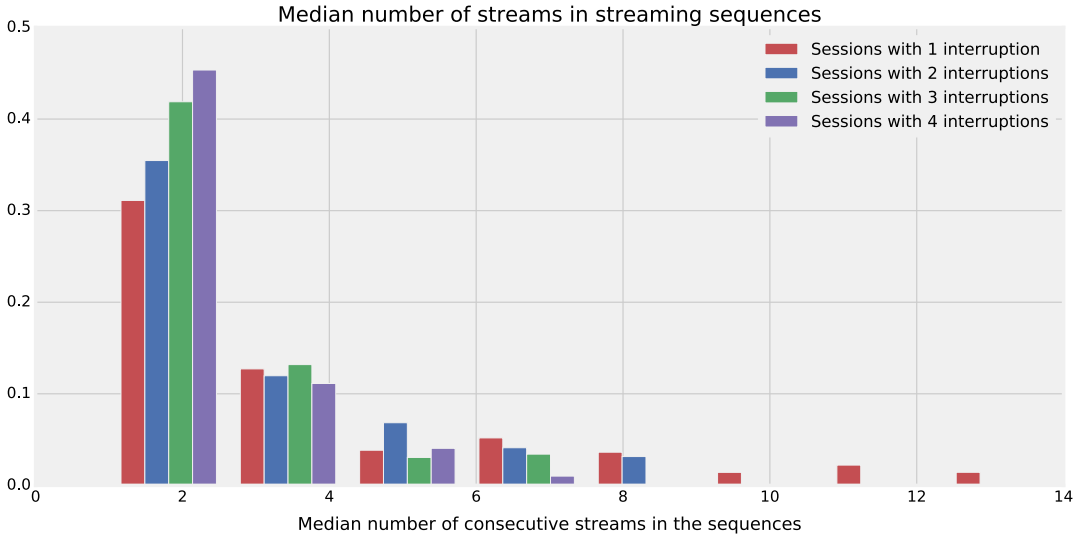
Furthermore, the structure and level of activity of these sections has a big impact on their length and the user engagement, as can be seen from the graph below:



Session length over all sessions is log-normal, but if we filter sessions by the number of sequences of interruptions, an interruption being a sequence of active events taken by the user interrupting a sequence of passive events. We observe that this number strongly correlates with session length.

Furthermore: the first two types of sessions (those with one or two interruptions) are bi-modal, which can be explained by the users either starting extremely short sessions (to listen to a specific song for instance) or playing music for significantly longer periods.

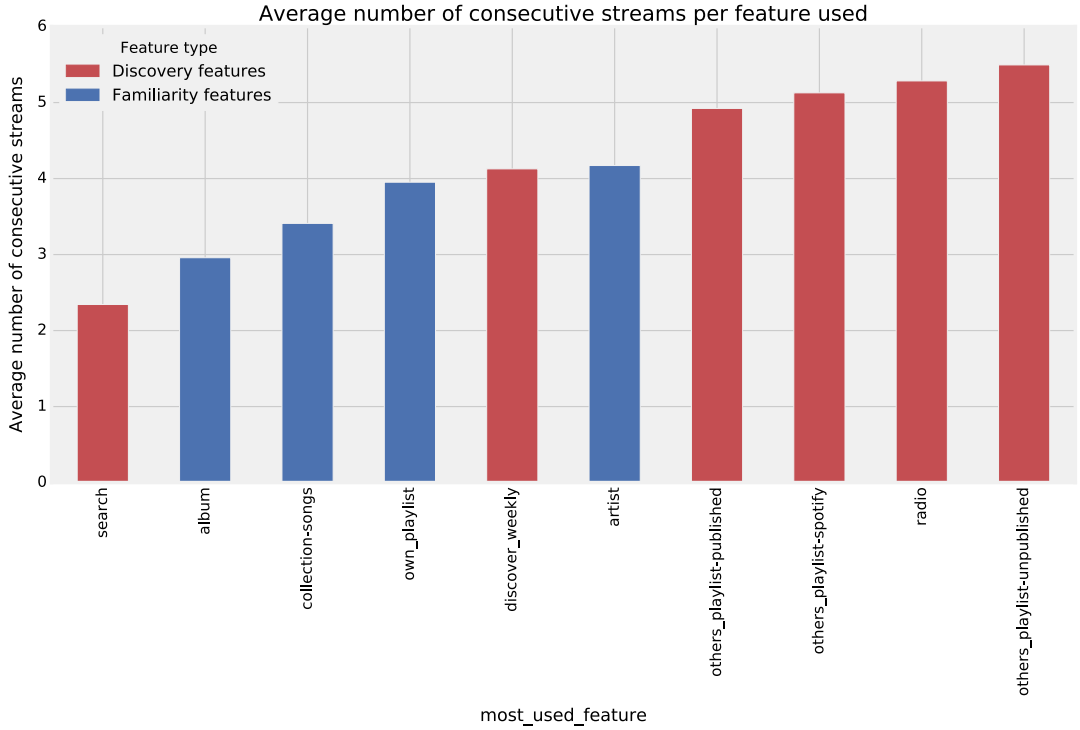
This is the case even if the total number of interruptions also negatively correlates with the size of consecutive streamed songs as can be seen in the graph below:



We notice that sessions with only one or two streaming sequences are the one that have a very large number of consecutive streams. These are more laid-back sessions where users loop

a large number of songs, and are naturally longer than those where users cherry-pick what they listen to.

The same type of analysis can be performed on a feature level to get a sense of how different features are used:



Other than the discovery vs familiarity issue we looked at previously, this shows that certain features encourage frequently interrupted listening whereas others are more lean back.

This is true for the search feature that is often used to play a specific song, and seldom to play a couple more than that, whereas other features like radio or playlists published by other users have longer successions of uninterrupted streams.

Simply analyzing sessions and streams from a binary perspective (active versus passive streams) forces us to miss the nuance in many of these analyses, and we need to build on this simplistic model to provide something more robust that allows us for instance to separate between a stream that happened right after an active event from the user, and a stream that happened five hours after that. This is what we will investigate in the next section.

5.3 Modeling user attention

We previously showed the type of patterns that can observed in streaming sessions. We also explained why taking into consideration user attention is vital for multiple types of analysis (A/B testing and recommender systems for example). In this section, we evaluate different models commonly used in similar use cases and propose a simple baseline model that provides deeper insights than simple frequency-based methods analyzed earlier.

Point processes

Given the previously described properties of the arrival of active events, one class of statistical models seems to be the most relevant: *self-exciting point processes*. Like noted before, the rate

of arrival of active events is not constant as we often have quick successions of active events followed by a long succession of passive events. In other words, active events have a higher likelihood of happening right after another active event than they do right after a passive one, and hence this process cannot be modeled as a homogeneous Poisson point process. (a similar observation was made for the arrival of sessions and streams in [31])

Examples of such self-exciting point-processes are the Cox Process, the Hawkes Process (commonly used in finance and contagion modeling [1]) or the Dynamic Contagion Process [33], a generalization of the previous two models.

An approximation for these models is to only consider the impact of the last point, instead of stacking the contributions of all events to calculate the likelihood of events to come.

Activity signals

A vital part of building a model for user attention is detecting user activity via the data collected from the clients. Multiple devices, such as smartphones and tablets, can provide explicit information about the user's level of activity (such as the device being moved, or the application being put in the background), but such signals are not collected by Spotify and are generally hard to analyze, especially when performing a cross-platform analysis. We chose instead to rely on the following signals which are simpler to collect and give us an explicit proof of the user's activity:

- The user opens the application
- The user skips a song
- The user chooses a song from a list
- The user seeks through a song
- The user closes the application

The user's navigation throughout the application (without necessarily changing songs) can also be seen as a proof of activity, but this was not included in this study for practical reasons.

Attention model

We choose to use the simpler excitatory model as a representation of users' attention level: every signal of user activity excites the user's attention, which decays with time until another excitation happens or user attention falls to a nil value.

This can be described by the following formula:

$$attention_t = e^{\frac{-(t-t_a)^2}{\delta_t}}$$

Where t_a is the timestamp of the last active event and δ_t is a coefficient that represents how slowly the user attention decays with time. We will call it decay coefficient in what follows.

We chose a decay factor of $\delta_t = 2250000s^2$, which means that the attention level falls to e^{-1} after 25 minutes.

This model is inspired by the more advanced contagion models described above, while still being easy enough to interpret and to implement. The exponential decay model is an approximation that yields enough information (especially compared to a simple binary measure) without having to dive deep in more complex models. These models are however definitely worth investigating, and we will discuss them in section 7.1.

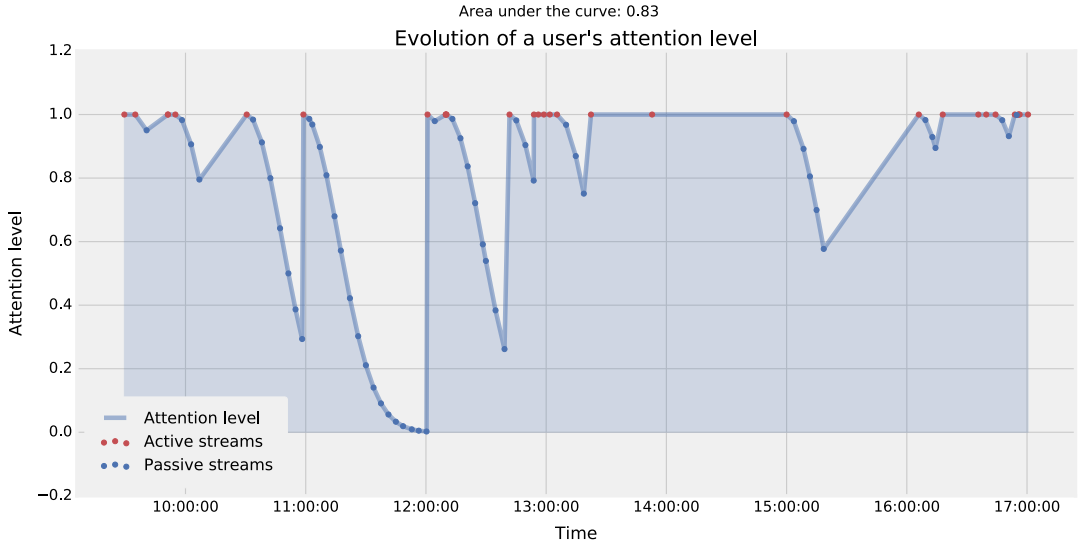
We use these events to tag streams as active (excitation) or passive (decay), and combine this with the model described above to calculate the attention level at every stream for all users and streams in our dataset.

Additionally, as a measure of how active a session is, we chose to use a similar approach to the one used in the previous chapter to aggregate stream polarity over a session: we integrate the attention level over the whole session, and normalize by dividing by the session length:

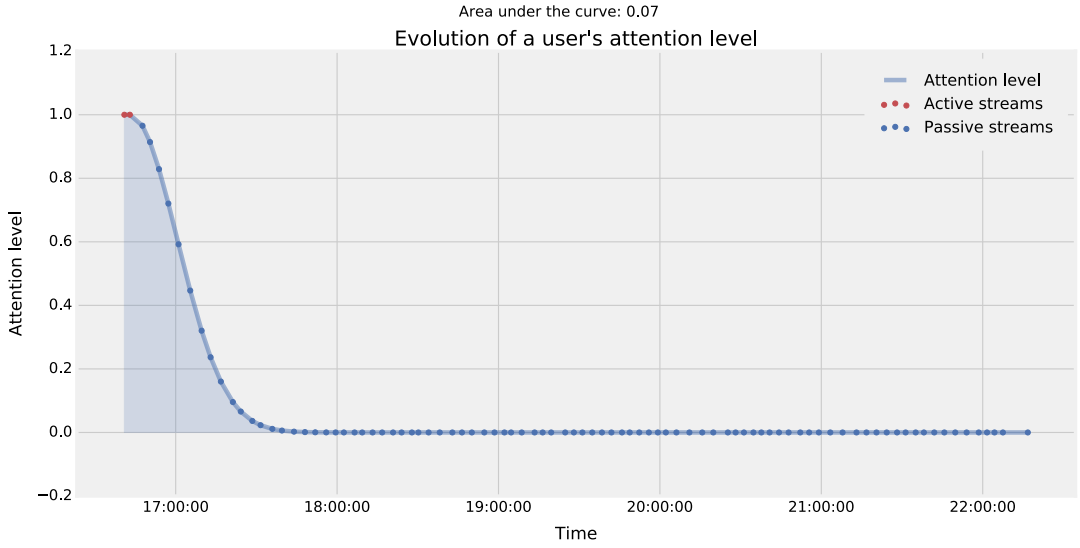
$$attention_{session} = \frac{1}{t_f - t_0} \times \int_{t=t_0}^{t_f} attention_t$$

Visualizing attention

Attention fluctuation throughout a session can simply be visualized thanks the the model described above. The result is a smooth attention curve that changes throughout the session depending on the user's actions (or inaction). As an example, here is the evolution of the author's modeled attention level during the 4th of November 2015, as he was listening to music and working on this report:



This model gives a visual way to analyze user sessions, and how active or passive this session is. The graph above shows three main sessions (one in the morning and two in the afternoon). The session in the morning starts rather actively, as the user has bursts of active events, but around 11:00 the streaming behavior goes into a more passive phase, with many songs playing one after the other as the attention level plunges to zero. The afternoon sessions are a bit more chaotic, with a certain number of active events but a very small number of songs that were streamed entirely.

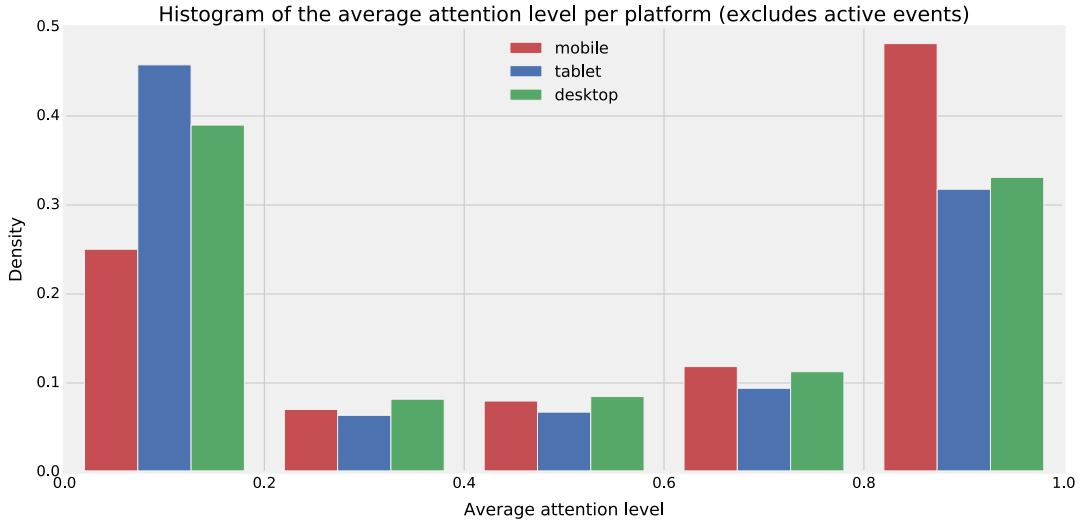


In contrast with the author's session seen above, this passive session starts with two active streams only to end in a long succession of passive streams. Streams starting one hour after the last active event are unlikely to gather the user's full focus. Their attention level is hence quasi-nil, and the session as a whole has a significantly lower area under the attention curve.

Attention monitoring

We think that using the area under the attention curve (that we will call attention integral) as a core metric for attention works well given the patterns observed in most user sessions. Since this metric can be calculated efficiently at scale and easily interpretable, this opens the door to a direct monitoring of how attention changes across sessions, users and features.

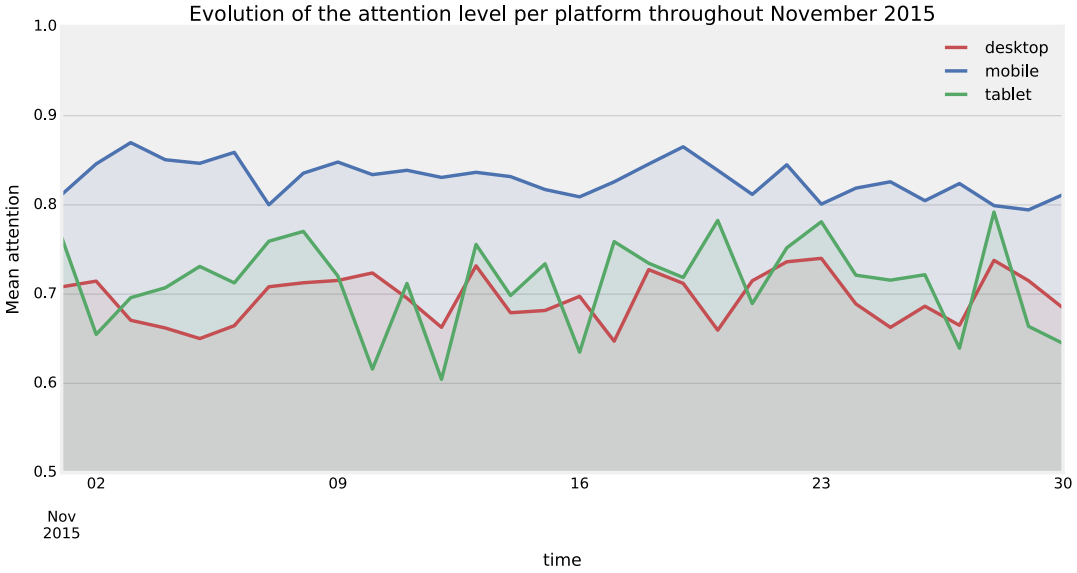
The per-stream attention level can also be used to evaluate the preponderance of active or passive behavior on the service. For instance, we can quickly get an idea about which platforms are the most commonly used in an active way:



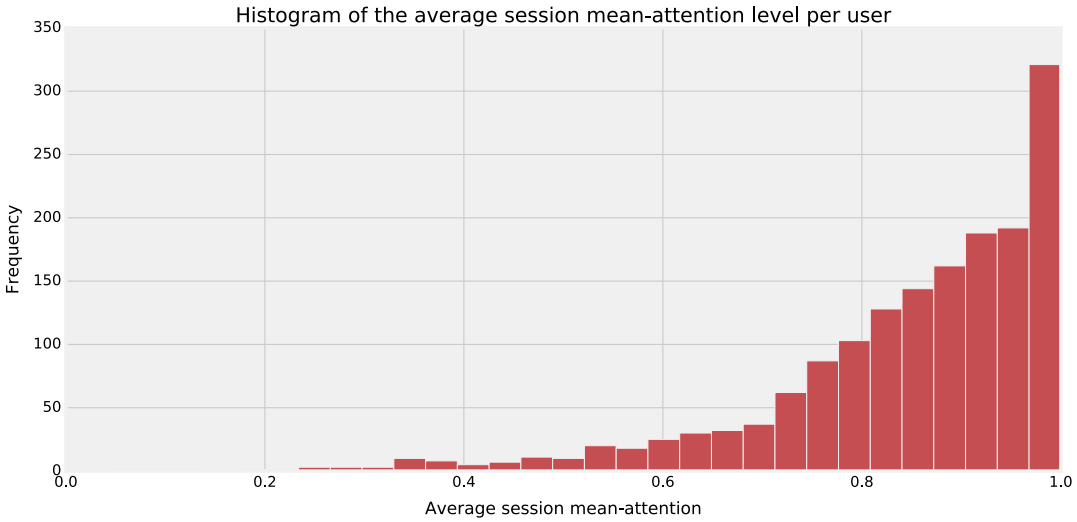
We notice a clear difference between the three types of platforms.

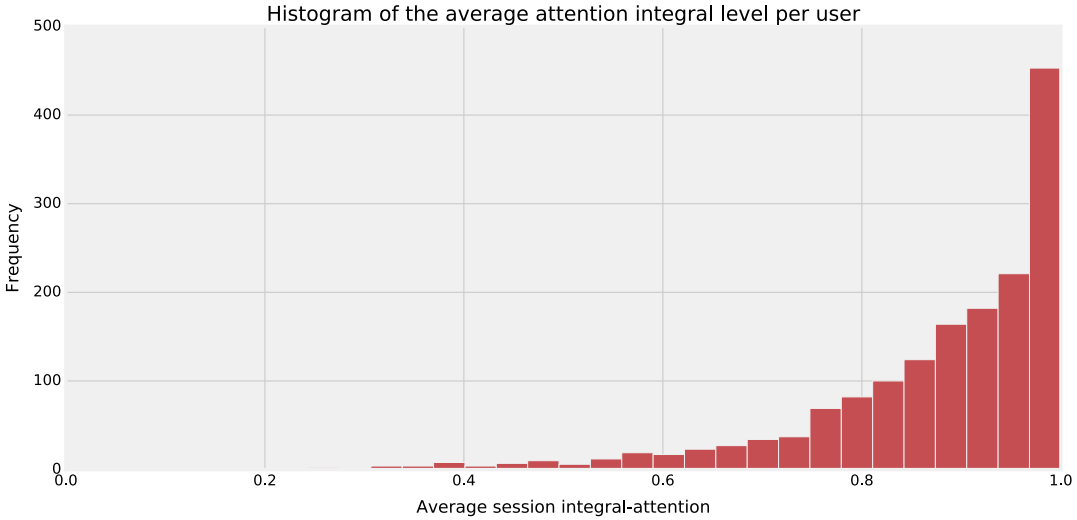
First, usage is significantly more active on mobile, where users usually handpick the songs they are listening to and often skip songs they dislike. Furthermore, tablet usage is the most inactive platform type. This is possibly because it is uncommon to use tablets with earphones to listen actively to music, and it is more common to use them with a dock station or external speakers to play background music.

The attention levels evaluated by our model can be tracked in real time, to detect any sudden changes (especially after the introduction of new features) that might indicate a change in user behavior or a malfunction in one of the clients.



In addition to these service monitoring applications, the attention level can be aggregated on a user level and used as an additional feature in the contextual model described in the previous chapter or used to customize the listening experience: More active users will tend to have a different behavior when it comes to music consumption, and the service should be tailored to fit this.



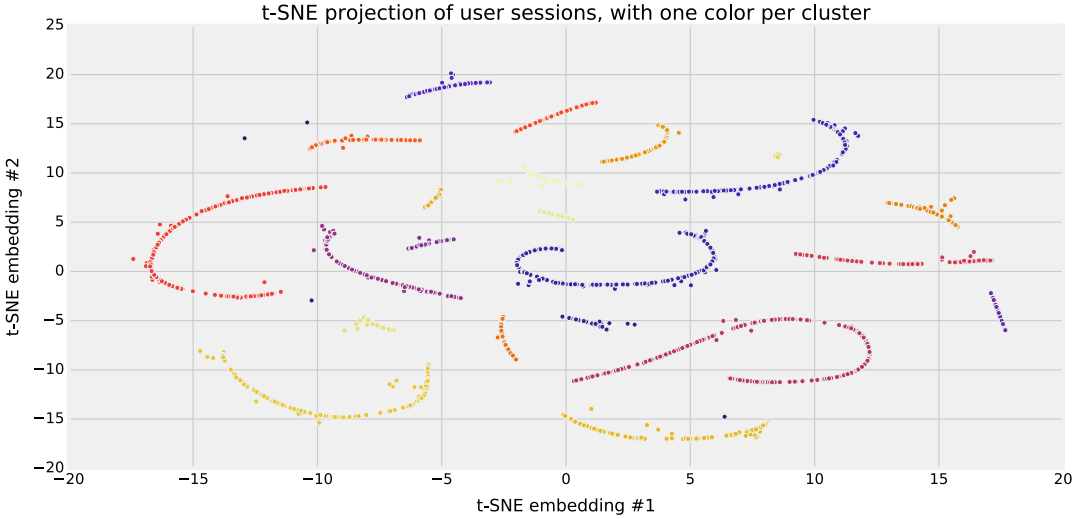


Visualizing session clusters

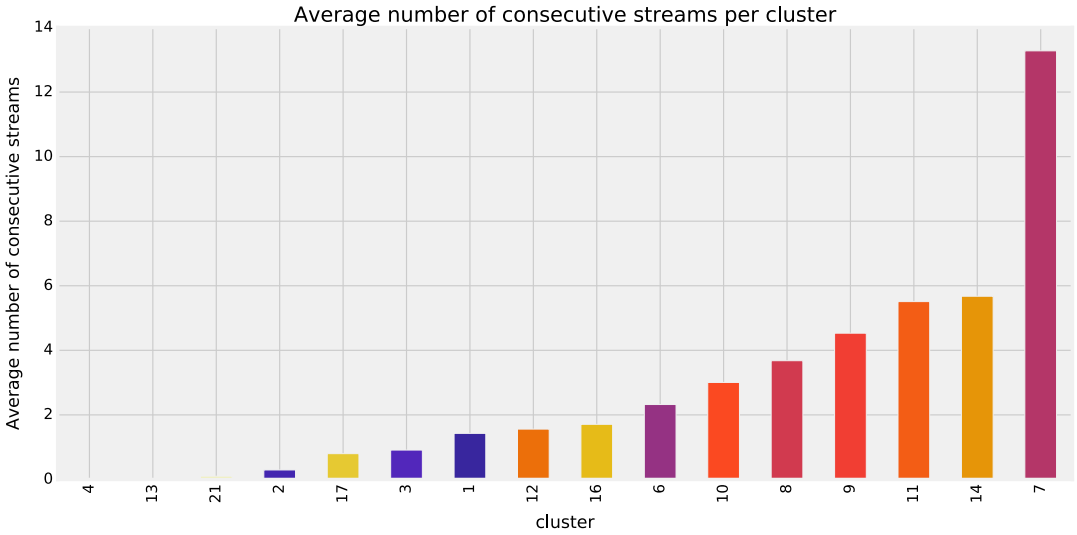
A more systematic to segment sessions is to use unsupervised clustering algorithms with the various attention-based features that can be extracted using our model, in addition with other general features (such as the number of playbacks and the skip ratio).

After aggregating session data, we end up with more than 20 features to describe every session. To visualize this (relatively) high-dimensional feature space, we use t-Distributed Stochastic Neighbor Embedding (t-SNE) [20], a technique that is commonly used to build a two or three dimensional embedding for high-dimensional datapoints. The main idea behind this technique is to minimize the difference in the probability distribution of the joint-probabilities between the embedding and the original vector space. In other words: points that are close to each other in the original space should still be close to each-other in the embedding.

We use this technique in conjunction to DBScan, a density-based unsupervised clustering algorithm, to segment the resulting embedding and visualize sessions:



Since most of the features used relate to the attention level, these is a significant intra-cluster variance on different attention-based variables:



This proposed clustering is simply a proof of concept, and many things remain to be done to obtain a meaningful segmentation of user sessions. For example, the string-like shape that session clusters have can be explained by the existence of highly correlated features that should be removed from the sessions' feature set. Additionally, other variables relative to the platforms and features used during the session should be taken into consideration.

5.4 Conclusion

After showing the great impact that attention has on user behavior, we presented a series of model that are appropriate to infer user attention from their actions on the service. We chose the simplest and most interpretable of these model, an exponentially decaying attention curve, and showed how it can be used to monitor the overall attention level on the service, with the example of platform monitoring, or even segment users and sessions.

This model remains overly simplistic, ignoring some user interactions (such as navigating the application without playing a song) and not taking into account nuances in the attention level depending on the action the user took, and the frequency of such actions. It can however easily be extended to include additional signals, and we have proposed more sophisticated models that evaluate a probability density using all events, instead of only using the last active event.

When it comes to automatically monitoring the levels of attention on a session level, the attention integral (area under the attention curve) gives a simple and interpretable metric to evaluate sessions based on their attention level. It is however heavily biased towards higher values, with a large proportion (59.91%) of sessions being in the $[0.95, 1.0]$ range.

This is mainly due to cases where a user is constituted of multiple active events separated by a long duration. Because we only calculate the attention level on every stream (and ignore views the user navigates through for instance), this sums up to a never decreasing attention during that period. Taking into consideration the whole timescale when calculating attention level (instead of only calculating it for streams) would solve this problem. Increasing the decay factor might also help for some other cases.

While it has a number of limits, we believe that this model allows a good first approximation of the fluctuations of user attention and has many concrete uses, ranging from measuring

the effect of a feature on attention through A/B testing to weighting implicit feedback for recommendation tasks.

Chapter 6

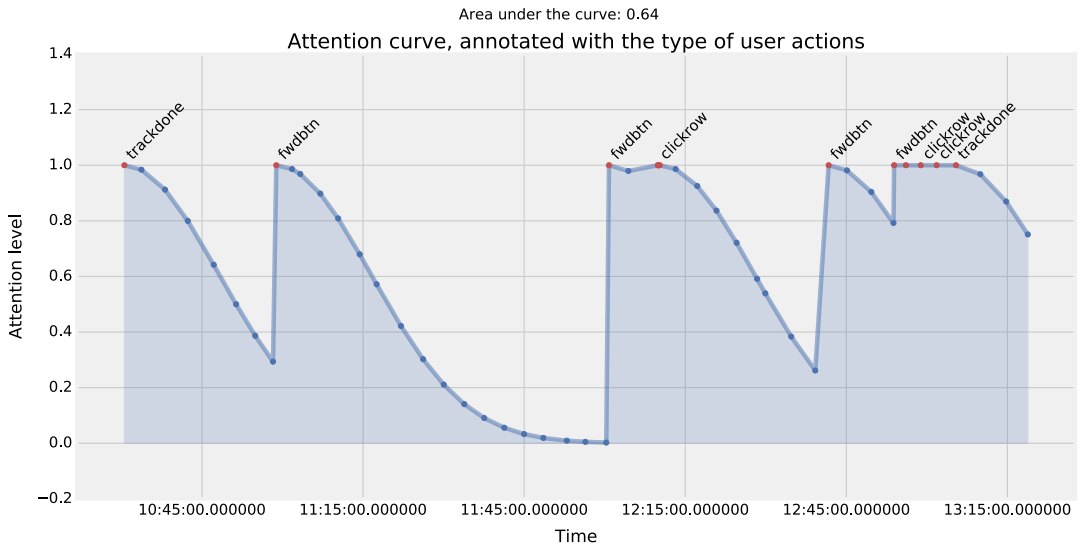
Sequential analysis and latent user states

One pattern in user behavior has been noted throughout the last two chapters: the arrival of user actions follow special chronological patterns coupled with the fact that the probability of having certain actions (such as skipping a song) changes drastically depending on the last action the user have taken.

In this chapter, we superficially explore this idea. We start by analyzing patterns in the succession of user actions and propose simple and interpretable sequential analysis models that can allow us to perform both an unsupervised exploration of user behavior and more traditional classification and prediction tasks.

6.1 Patterns in user actions' arrival

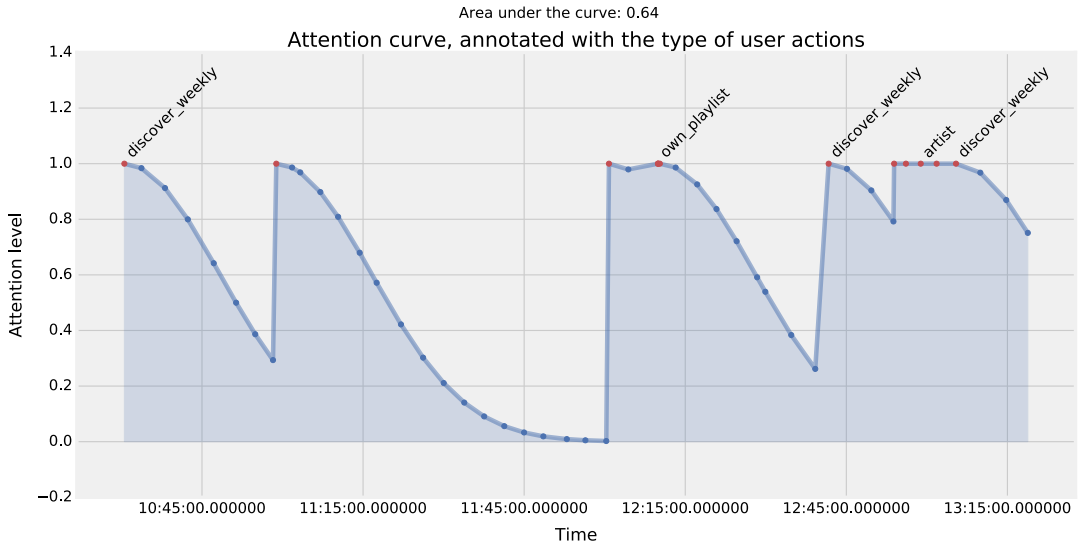
Before choose a model for user actions, we have to analyze the type of behaviors and patterns that can be observed in user sections. We can do this by visualizing and analyzing a session we have seen in the previous chapter:



As noted in the user attention chapter (chapter 5), the first thing we notice is the user alternates between passive and active sequences, where the passive sequences are sequences

where the user simply finishes one track after the other (the `trackdone` event), and in the active ones the user either chooses a song from a playlist (the `clickrow` action) or skips a song (the `fwdbtn` action). Additionally, there are some more punctual active events that interrupt a long passive sequence but are directly followed by a passive sequence.

We can apply the same analysis by following the change in features used to stream music:



This sessions shows some feature transitions that are common among Spotify users: Discover Weekly is (as its name indicate) often used to discover new songs and artists. This sometimes leads users to go to the artist page of a newly discovered artist (transition to the `artist` feature) or save some of the songs they discovered to one of their playlists (transition to `own_playlist`).

Our hypothesis is that these user actions can be modeled as a stochastic process affected by latent variables that define the state of a user. We will start with a simple stochastic model for user actions' transitions, and then move to a latent model and try to extract different states users can be in.

6.2 Markov Chain analysis

Markov Chain is a stochastic process that models the transition between a set of states. The main particularity of this model is that it is memoryless: in other terms, the probability of moving to a state only depends on the last state, and not on any of the previously occupied states. This property is called the Markov property.

Markov Chain are very commonly used in a series from fields among which thermodynamics, speech recognition, business analytics and even sport modeling. Even if the Markov property can be limiting for more complex processes, this model is usually a good baseline to model dependencies in sequential events.

Every Markov Chain can be described by a set of states and a transition matrix that describes the probability of moving from a start state (row) to a destination state (column). In other terms, the Markov Chain is entirely defined by a matrix M such as:

$$M_{i,j} = P(state_{t+1} = j \mid state_t = i)$$

It can happen that the marginal likelihood for the destination state is not significantly different from the conditional transition probability:

$$P(state_{t+1} = j \mid state_t = i) \approx P(state_t = j)$$

If that is the case, the transition matrix has a rank of one and the random process is a simple sampling from a categorical distribution. As such, Markov Chain is not an appropriate model to use in this scenario.

We start by calculating the transition probabilities and compare them with the marginal likelihoods to confirm that there are strong dependencies in the transitions from one user action to the other. We filter out events that happened less than 300 times in our dataset, as those are most often anomalies:

reason_start	reason_end	number	transition_probability	marginal_likelihood
trackerror	trackdone	1915	0.707	0.577
	fwdbtn	495	0.183	0.257
trackdone	trackdone	305339	0.841	0.577
	fwdbtn	36950	0.102	0.257
	clickrow	19954	0.055	0.160
	appload	316	0.001	0.004
popup	popup	435	0.891	0.001
playbtn	playbtn	696	0.830	0.001
fwdbtn	trackdone	34020	0.203	0.577
	fwdbtn	125107	0.745	0.257
	clickrow	8568	0.051	0.160
clickrow	trackdone	45740	0.337	0.577
	fwdbtn	11006	0.081	0.257
	clickrow	78792	0.581	0.160
appload	trackdone	7859	0.550	0.577
	fwdbtn	2190	0.153	0.257
	clickrow	1887	0.132	0.160
	appload	2355	0.165	0.004

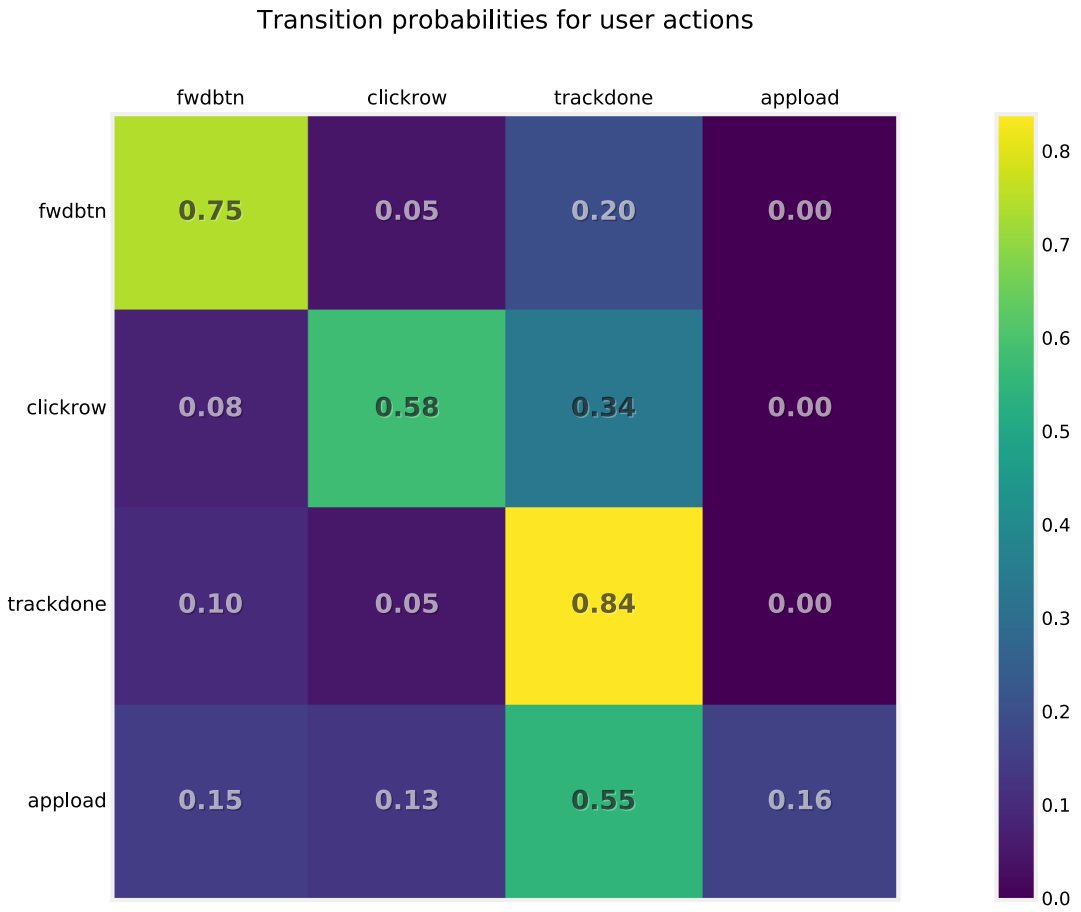
We observe large deviations between the marginal likelihood of some events and their conditional transition probability. For example, regardless of the last action a user took, he has a 57.68% chance of finishing a song (`trackdone`) action. However, if we know that the last action they took is also a `trackdone` action, this probability increases to 84.14%. The difference is even higher for skipping songs, where a user only has an overall 25.68% chance to skip a song, but this drastically increase to 74.51% if the last action they took was also a skip. Overall, the RMSE (root-mean-square error) between these two likelihoods is of 35.75%.

This can be explained by the fact that the cost to skip a song, especially on mobile platforms, is lower after a first skip was done. These can either be for mechanical reasons (needing to take the phone out of one's pocket, open the application and scan the interface for the next button) or simply a higher level of attention that makes the user more aware of what they are listening to, and hence more likely to skip a song.

This also explains the high importance features based on `reason_start` had after fitting machine learning models to predict skipping behavior.

As many actions, such as `popup` or `playbtn`, rarely occur and are uninformative of user behavior, we restrict ourselves to four main actions: $\{`fwdbtn`, `clickrow`, `trackdone`, `appload`\}$

The transition matrix for these events can be seen below:



The `appload` action is special because it is only emitted when the application starts, and as such it is natural that users cannot move back to this state.

We notice that the highest transition likelihoods are those that keep the user in the same state. This confirms are hypothesis linked with the cognitive (and mechanical) cost of doing a different action: when a user is in a passive listening session, it takes them more energy to switch a song than if they were already skipping through a playlist.

We also notice that both `fwdbtn` and `clickrow` have a reasonable likelihood of leading to a finished song (`trackdone`). However, `clickrow` has a higher likelihood of leading to a finished song. This could also be caused by the difference in interactions that lead to these actions: `fwdbtn` can be done by clicking on the next or back buttons, but also using easily accessible elements integrated to some devices and operating systems. However, this might only account for a small difference, whereas the difference we observe is significantly higher. We believe that this is caused by the fact that users consciously choose the next song to play with `clickrow`, whereas they rarely know what song will be played next when performing a `fwdbtn` (because they are browsing through a playlist and/or using shuffle mode). Since users are naturally more likely to listen to a song they chose than one chosen randomly, this is naturally reflected in a higher transition probability from `clickrow` to `trackdone`.

If this analysis allows us to make general observations about actions taken by users, it does not describe all the complexity behind these actions: given the environment in which they stream music, their current mood and their goals using the app, they will tend to exhibit radically different behaviors.

6.3 Latent user states with Hidden Markov Models

A Hidden Markov Models is a statistical model that can be used to model systems that generates observable emissions, but where the likelihood of observing an emission depends on which unobservable state the system is in. The hidden states in a Hidden Markov Model are a Markov Process: a transition matrix describes the probabilities of moving from one state to the other. In addition to this, every the hidden states function like latent variables in topic models, as every state has a different distribution of emissions, and different models can be used to represent this behavior. For continuous emissions, such as the amplitude in a spectrogram or the coordinates of some pixels, a Gaussian distribution can be used. If a simple Gaussian does not fit the data well enough, it can be substituted by a Gaussian mixture model. For discrete emissions, such as phonemes in speech recognition or a set of actions like in our use case, we use a multinomial model where every state has a categorical distribution over all possible discrete emissions.

Hidden Markov Models are more complex than they seem, and they have been successfully applied to challenging problems in cryptography, speech recognition and even activity recognition in the field of robotics [26].

HMMs can be used to classify sequences of actions describing different classes. In this use case, an HMM model is trained for every class from a training set, and sequences from the test set are labeled by picking the model that has the highest likelihood of having generated that sequence. This likelihood estimate can be efficiently calculated using the "forward algorithm", based on dynamic programming principles.

These models can also be used in a generative fashion: by learning the rules governing some stochastic process, we can sample any desired number of emissions following the same rules to produce similar results. For example: an HMM trained to recognize a certain sound can be used to synthesize this sound.

The parameters that this model fits can also be used in an exploratory way, to extract insights about the structure of the data and the latent states responsible of the observed behavior.

Training a Hidden Markov Model on user actions

We want to train a Hidden Markov Model on the user actions taken in the large number of user sessions at our disposal. We ignore the `appload` action because it is always emitted at the beginning of a session, which is useless in the case of an HMM model since it already calculates a set of initial state probabilities. We add a virtual `session_end` action which represents the end of a session. The goal behind this is to see if there are any correlations between certain latent user states and the desire to finish a session.

An HMM only requires a series of observed emissions to learn the hidden states' transition matrix, the parameters of states emission and the initial hidden state probabilities. This is done by following the Baum–Welch algorithm, a special case of the EM (Expectation Maximization) algorithm. This procedure is unsupervised and does not require any labels for the given sequences of emissions, however we need to supply an important parameter to the model: the number of hidden states.

The number of hidden states of an HMM model has a high impact on its use, the resources needed to fit it to the data, its interpretability and the complexity (and risks of overfitting) associated with the model. A higher number of states guarantees a higher likelihood of generating the training data. However, after a certain number of states, the gains are marginal and we end up with a number of redundant states that have similar emission distributions. Additionally, a higher number of states means a lower interpretability for the model, a slower training process and a higher risk of overfitting as the number of parameters increases quadratically with the number of hidden states.

As such we must not only aim to maximize the likelihood but also introduce a regularization parameter to limit the complexity of the model. Many methods are used to find the optimal number of hidden states for a given dataset, most often methods based on measuring the log-likelihood of a test set using cross-validation, combined with a measure of the complexity of the dataset such as AIC (Akaike information criterion) and BIC (Bayesian information criterion) [5].

We used a simple evaluation approach based on these principles, by training our model on 60% of the user sessions and calculating the log-likelihood over the 40% remaining sessions. We also calculate the AIC of every model and normalize it to get an idea of the relative balance between complexity and likelihood maximization.

The AIC metric can be defined as follows:

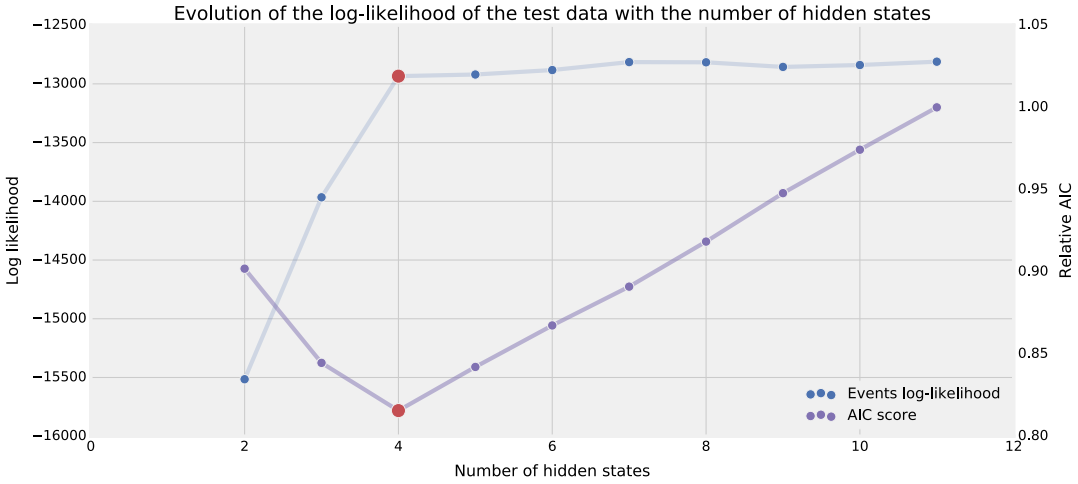
$$AIC_{model} = 2 \times \|parameters\| - 2 \times \log(likelihood)$$

In the case of a multinomial HMM, the number of parameters to fit is (with h the number of hidden states and e the number of emissions):

$$\|parameters\| = M_{transitions} + M_{emissions} + V_{initial\_probabilities} = h \times h + h \times n + h$$

Given the difference in scale between the log-likelihood and the number of parameters, a normalization constant is often introduced to wait these two components. Since we will evaluate our model on a large number of sequences, the absolute value of the log-likelihood has a significantly higher value. We decided to weight it with a 0.002 factor. We chose this factor because it gave an impact to both parameters, resulting in an informative plot (unlike higher values that totally neglect the complexity of model, and vice versa).

The results of this evaluation can be seen below:

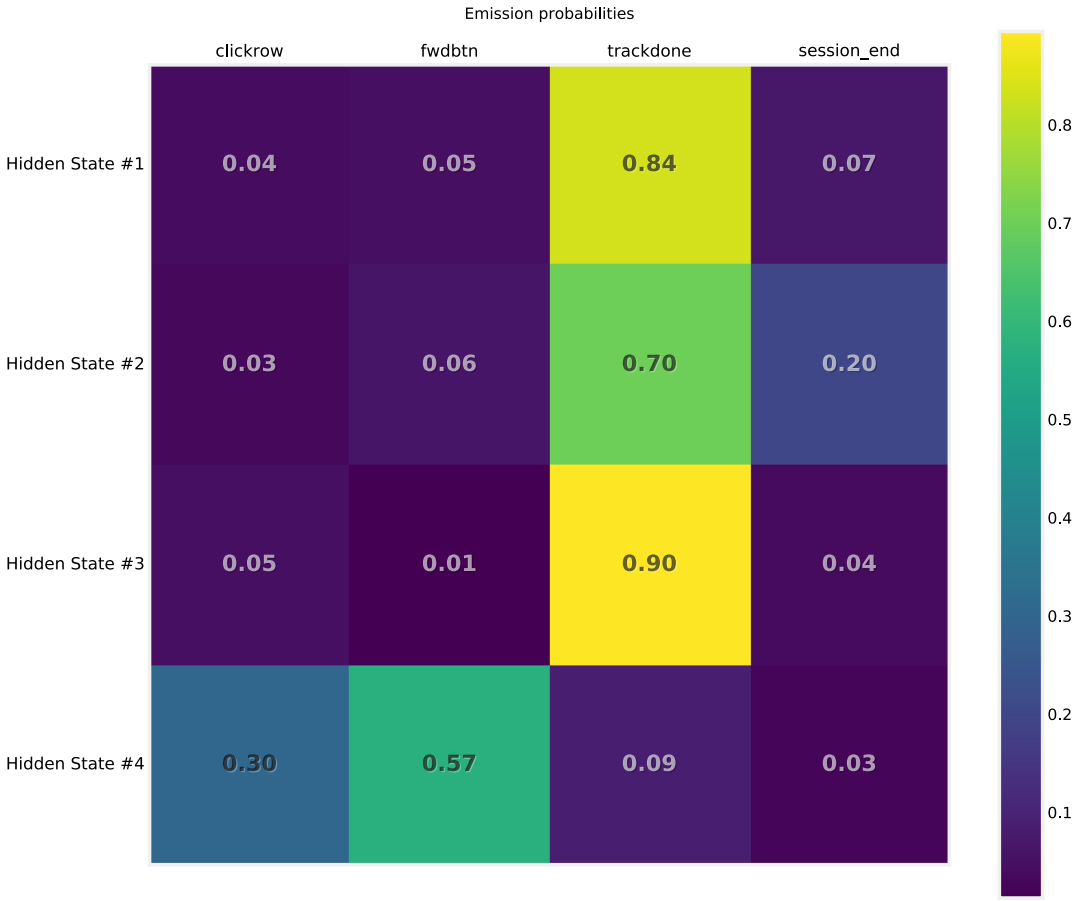


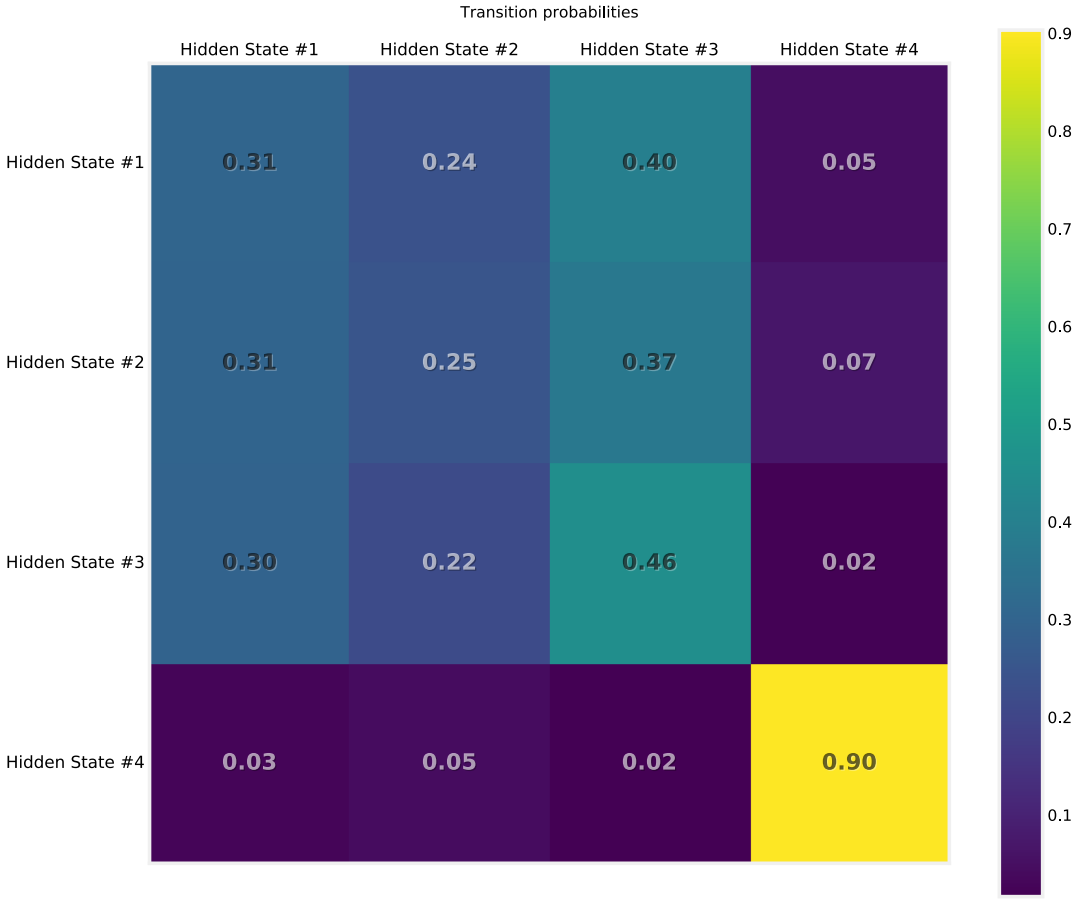
By using the log-likelihood estimate and following the "elbow method", choosing the point where the curve's angle gets lower than a certain threshold, the optimal number of states is 4. This is confirmed by the relative AIC scores that has an inflection point at the same value, meaning that the balance between model complexity and likelihood maximization is achieved at that point.

Interestingly enough, the number of emissions we model is also equal to 4. If the HMM we fit has one predominant emission per state, we revert to the case of a Markov Chain and using an HMM is unnecessary.

Now that we chose the optimal number of states, we fit a multinomial HMM to the entirety of the data to visualize the emission and hidden state transition probabilities and use it to predict users' latent states.

Application of a Hidden Markov Model





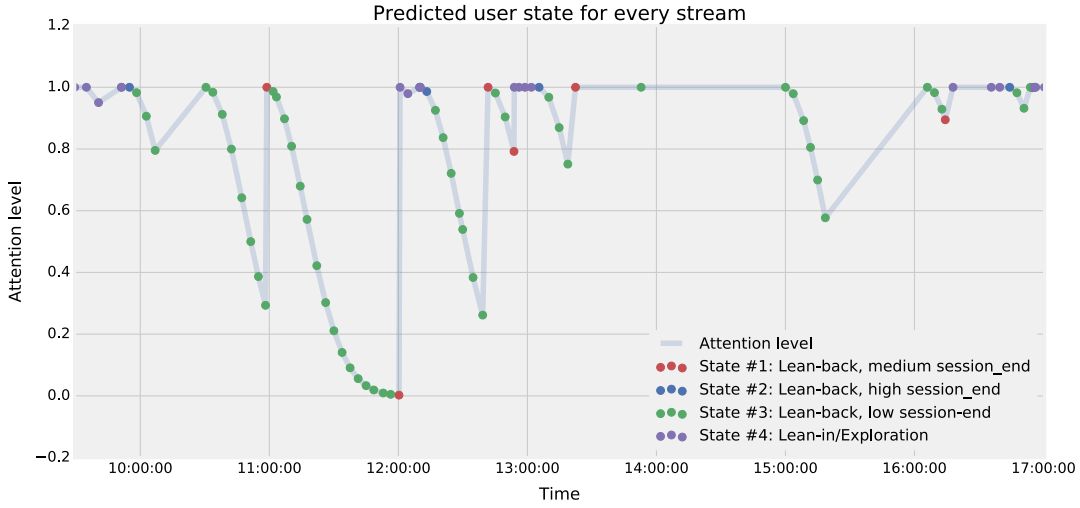
Because the hidden states have a diverse distribution of emissions in a couple of hidden states, the HMM provides a real added value compared to a Markov Chain.

Interestingly enough, instead of having one latent state where skipping (`fwdbtn`) is predominant and one latent state where picking songs is predominant (`clickrow`), these two actions are both amalgamated in the second latent state which could be labeled as the "discovery" latent state: the state that the user is in when they navigate through songs and playlists. These two actions are almost interchangeable in this state, with a higher likelihood for `fwdbtn` as it is more common. We should note that the probability of finishing a track in this state is far from being nil: this observation, in addition to the fact that users are usually "trapped" in this state, can mean that users while being in a discovery-oriented state will seldom listen to an entire song before getting back to skipping and picking other songs.

The third hidden state has a very high probability of finishing a song (`trackdone`). But it has an equally low probability to finish a session as the more discovery-oriented hidden state. However, two other latent states also have a high probability of finishing a song and a low likelihood of skipping or selecting songs, with various likelihoods of ending the session. We think that this might simply explain the difference between lean-back and lean-in listening: when the user is playing one song after the other with a low likelihood of moving to a more active/exploratory state, they are less likely to end the session. This it is the case in the third hidden state that has the lowest likelihood to move to the exploratory state (the fourth hidden state) and also the one of the lowest likelihoods ending a session. Inversely, the two other "passive" latent states have a higher likelihood of moving to the exploratory state, and this likelihood is correlated with their likelihood of ending a session. In other terms: after listening

to a series of songs, the user's likelihood to skip a song is correlated with them ending the session.

In addition to this unsupervised analysis, we can use the model that we trained to predict the latent states a user was in during a session:



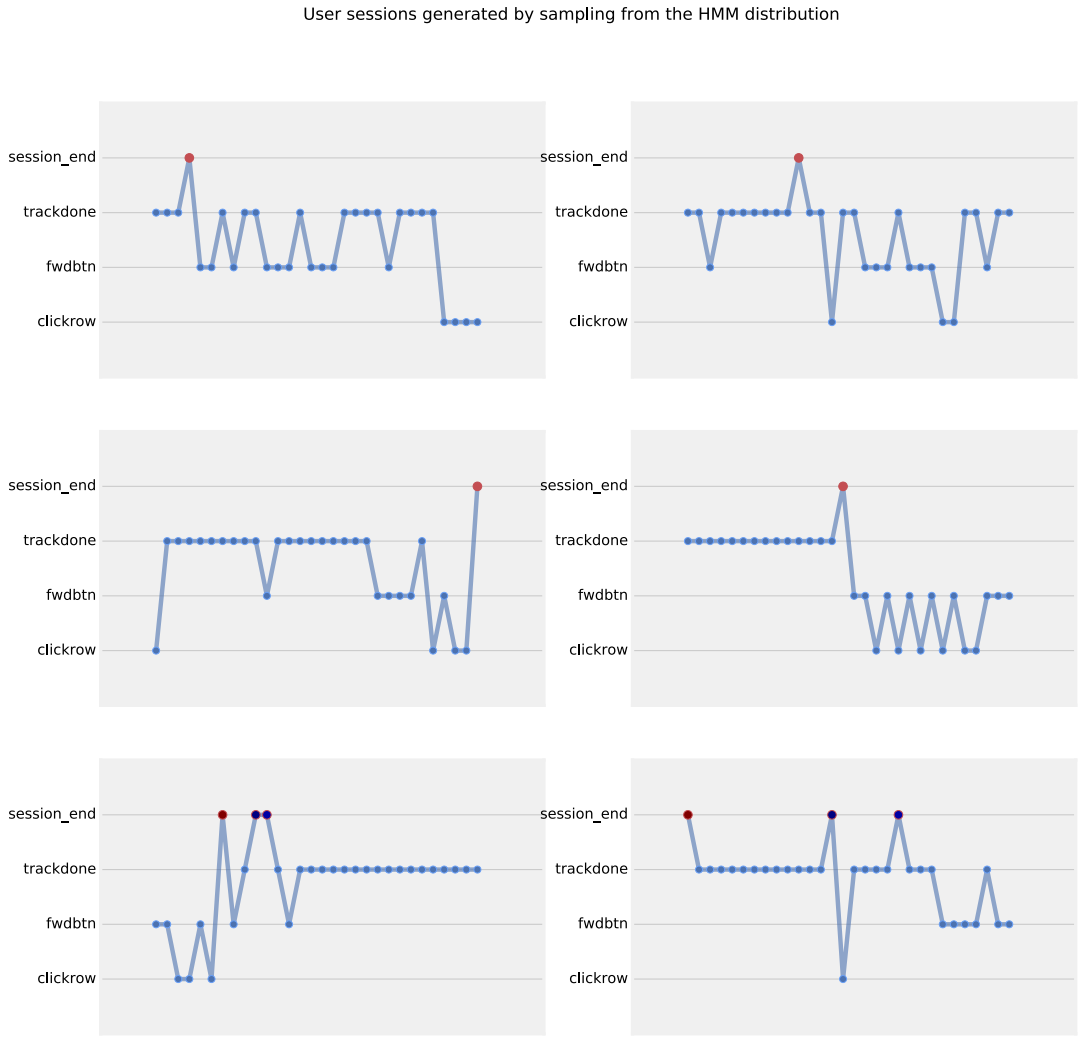
These state predictions can be used as additional features to characterize user sessions: a session where the user was mostly in a lean-back state is radically different from what that is mostly exploratory.

Since these predictions can be efficiently calculated (with the forward algorithm), they could even be used online while the user is interacting with the system to customize his experience or the click-through rates on advertisement (commonly used for free streaming users): if a user is in a state where they are likely to end a session, we might want to avoid launching an advertisement. Similarly, if a user is in a more lean-back session, we could avoid presenting advertisement that requires his attention. Not only can we predict the most probable latent user state, but we can also predict the most likely user action. This as well can have multiple applications in adapting the user experience or providing them with useful hints.

The usefulness of these latent state predictions however depends on the accuracy of such a model, which is hard to ascertain without a rigorous evaluation process using labeled examples.

Additionally, one important use case for Hidden Markov Models in the context of streaming services is user classification and segmentation: given different classes of users (such as: churning user versus converting user; or fraudulent versus legitimate users), we can identify special patterns in their behavior by training one HMM per user class, and evaluating the class of a user by calculating the log-likelihood on each of these models and picking the best performing model. The same can be done on an even more detailed level by classifying sessions. An interpretative analysis of the fitted parameters (such as the one we have done at the beginning of this section) can also be useful to understand the behavior of these different classes of users.

Finally, one more questionable application for the HMM model is its generative property: we can use an HMM trained on different types of users or sessions to generate new sequences of user actions. While this might not be the most useful of applications, it can be helpful to interpret HMM models that have a high number of hidden states, as the human brain is more used to detecting patterns in graphs than deciphering high-dimensional matrices.



6.4 Conclusion

In this chapter, we motivated the need for a sequential analysis of user actions on platforms such as Spotify, and presented two models commonly used for this task.

A simple Markov Chain model allows us to get general insights on these actions, such as the fact that users alternatively get stuck in passive or active states.

Hidden Markov Models allow a more fine-grained analysis, and have multiple applications: unsupervised exploration of the logic behind user actions, predicting users' latent states and segmenting their sessions, classifying users and detecting unexpected behavior and potentially adapting the user experience according to a user's latent state.

However, it is questionable whether this analysis can lead to immediate actionable results for Spotify, especially since there are only subtle difference between many of the latent states learned by the HMM model. These models can only be used if a rigorous evaluation process is put into place.

Chapter 7

Conclusions

The goal of this project was to explore under-exploited aspects of user behavior in music streaming services and show how they can be used to improve these services and infer their users' preferences and sentiments more accurately.

The first aspect we explored is streaming context. We showed how contextual variables such as the type of device or the action used to start a song has a large effect on its outcome. Building on this observation, we proposed a model that evaluates the importance of every one of these features using machine learning techniques. Using a similar approach, we leveraged the predictive power of these models on user skipping behavior to infer what we call "stream polarity": a weight that ranges from -1 to +1 and expresses to which extent an action was positive or negative, giving more weight to unlikely actions as they give more information about the user's preference for a specific song rather than the effect of their context. We showed how aggregating this polarity over multiple dimensions, such as users, can provide a single expressive metric to represent user satisfaction, and showed that there is a strong correlation between this metric and business metrics such as user conversion and churn.

The second concept we focused on is user attention. After explaining the "attention economy" and its relevance to streaming services, we performed a preliminary analysis of its impact on user behavior on Spotify. We then proposed different models for inferring the level of attention and focus of a user, settling on the simpler exponential decay model. This model is a good first approximation, but as we will see in section 7.1, it ignores aspects that are modeled by more sophisticated point-process models. We proposed different methods to aggregate this modeled attention level, and evaluated their application on tasks such as separating more "lean-back" and passive listening from active behavior by using clustering algorithms.

We also motivated the relevance of sequential analysis methods in the context of streaming services. These methods take into consideration the succession of user actions and the dependencies between different actions that can be hard to detect by approaches that ignore the chronological aspect of the data. By fitting a Markov Chain on user actions used to end a stream, we showed that they are strongly linked and that some patterns, such as long successions of skipped songs, are common among all users. We used a Hidden Markov Model to explore the idea of latent user states that impact their behavior on the service. We showed how this model can be used in an unsupervised way to infer a structure in user actions, but also in a predictive way to do fraud detection or classify users.

The methods introduced in this project can be used as-is to study specific aspects of streaming services and improve them: monitoring changes in user satisfaction, the evolution of user attention and engagement with the introduction of new features, finding patterns in streaming behavior to improve the user experience.

One particularly promising application is the combination of the user attention and streaming context models to improve the implicit feedback given to recommender system as these models counter some of the frequently cited issues with implicit feedback, such as its inherent

noise and the lack of negative feedback.

7.1 Future Work

The goal if this project was to explore different facets of user sentiment and behavior and propose relevant models for these components. As such, we had to evaluate a number of themes and concepts often restricting ourselves to a shallow analysis and simple models.

One example of this superficiality is the attention level model we chose: if an exponential attention decay is a good baseline to analyze user sessions and the fluctuation of attention on the platform, it is overly simplistic and misses on many nuances, most importantly the combined effect of multiple active events in contrast with a single action. The next step to build a more robust and comprehensive model would be to experiment with self-exciting point processes that we presented in that chapter, such as the Hawkes Process [1] or the Dynamic Contagion Process [33]. These models have proven their worth in many fields (like finance and health-care) and are definitely worth investigating.

In the user attention chapter, we quickly explored the idea of session clustering. This was done as a proof of concept and lacks some rigor in comparison with other parts of the same chapter: certain correlated features degrade the quality of the clustering and other important features (such as those related with platform and streaming feature usage) are missing.

The same remark can be done about the sequential analysis chapter: we showed the importance of a sequential analysis of user actions and proposed a series of models based on Markov Chains. However, we did not develop the full potential of these models and some of their possible applications because of our limited time.

Another master thesis student is already working with Spotify's analytics department on both of these issues by focusing on user sessions. This includes finding better features and methods to cluster sessions and applying the Hidden Markov Models not only on user actions but also on other variables such as the platforms used or features used by a user. Another promising use for Hidden Markov Models is fraud detection, which is increasingly important for platforms like Spotify as attempts have been made in the past to attack the system with bots.

Our model for stream polarity and user sentiment has many applications on systems that are vital for streaming services like Spotify and it can be generalized to other uses involving implicit user feedback as well. The results are promising, but there is a major caveat that needs to be addressed when researching this further: we need to run these experiments on a significantly larger dataset, covering a longer period of usage, to have a higher confidence of the relevance of the polarity score, especially when it comes to predicting churn and conversion.

We discussed recommender systems to a great extent in the related work section because our hypothesis was that building a model for user sentiment and stream polarity in addition to taking into consideration user attention would be a great improvement over the methods currently in place, which basically amounts to considering all streams equal. It is therefore important to evaluate the usage of these two models as a way to enrich the input to recommender systems and improve their performance. We started developing `receval`[1], a framework to facilitate this evaluation, inspired by the Rival framework [28]. But since the scope of this project gradually drifted away from recommender systems, we abandoned this effort. We however think that this is the logical next step, and that there remains plenty to be done to extract signal from the inherently noisy implicit feedback.

[1]The code source for this framework can be downloaded online: https://github.com/halflings/receval

Bibliography

[1] Emmanuel Bacry, Iacopo Mastromatteo, and Jean-Francois Muzy. Hawkes processes in finance. page 48, 2015.

[2] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal ofMachine Learning Research*, 13:281–305, 2012.

[3] Leo Breiman. Bagging Predictors. *Machine Learning*, 24:123–140, 1996.

[4] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. Classification and regression trees belmont. *CA: Wadsworth International Group*, 1984.

[5] Gilles Celeux and Jean-baptiste Durand. Selecting Hidden Markov Model State Number with Cross-Validated Likelihood. 2008.

[6] Tianqi Chen and Carlos Guestrin. XGBoost : Reliable Large-scale Tree Boosting System. pages 1–6.

[7] D.R. Cox and E.J. Snell. *Analysis of Binary Data, Second Edition*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1989.

[8] T.H. Davenport and J.C. Beck. *The Attention Economy: Understanding the New Currency of Business*. Harvard Business School, 2002.

[9] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[10] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: Evaluating Recommender Systems by Coverage and Serendipity. *Proceedings of the fourth ACM conference on Recommender systems - RecSys '10*, page 257, 2010.

[11] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. Characterizing user sessions on YouTube. *Proceedings of SPIE*, 6818:681806–681806–8, 2008.

[12] Carlos A. Gomez-Uribe and Neil Hunt. The Netflix Recommender System. *ACM Transactions on Management Information Systems*, 6(4):1–19, 2015.

[13] Asela Gunawardana and Guy Shani. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *The Journal of Machine Learning Research*, 10:2935–2962, 2009.

[14] Ahmed Hassan, Xiaolin Shi, Nick Craswell, and Bill Ramsey. Beyond clicks: Dwell Time for Personalization. *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13*, pages 2019–2028, 2013.

[15] Shuk Ying Ho and David Bodoff. The effects of personalization and familiarity on trust and adoption of recommendation agents. 38(2):497–520, 2014.

[16] Yifan Hu, Florham Park, Yehuda Koren, Chris Volinsky, and Florham Park. Collaborative Filtering for Implicit Feedback Datasets. 2008.

[17] Polonetsky Jules and Omer Tene. Privacy in the Age of Big Data: A Time for Big Decisions. *Stanford Law Review Online*, 64:63, 2012.

[18] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):42–49, 2009.

[19] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[20] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[21] Wes McKinney. Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, 1697900(Scipy):51–56, 2010.

[22] S M McNee, J Riedl, and J a Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. *CHI'06 extended abstracts on Human factors in computing systems*, page 1101, 2006.

[23] Denis Parra and Xavier Amatriain. Walk the Talk: Analyzing the relation between implicit and explicit feedback for preference elicitation. In Joseph A Konstan, Ricardo Conejo, José L Marzo, and Nuria Oliver, editors, *Proceedings of the 19th International Conference on User Modelling Adaptation and Personalization (UMAP 2011)*, volume 29, pages 255–268. Springer-Verlag, 2011.

[24] Denis Parra, Alexandros Karatzoglou, Xavier Amatriain, and Idil Yavuz. Implicit feedback recommendation via implicit-to-explicit ordinal logistic regression mapping. *CEUR Workshop Proceedings*, 791(1), 2011.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] Alessandro Pieropan, Karl Pauwels, and Hedvig Kjellström. Audio-Visual Classification and Detection of Human Manipulation Actions. (Iros):3045–3052, 2014.

[27] Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. Classifying latent user attributes in twitter. *Proceedings of the 2nd international workshop on Search and mining user-generated contents - SMUC '10*, page 37, 2010.

[28] Alan Said and Alejandro Bellogín. RiVal – A Toolkit to Foster Reproducibility in Recommender System Evaluation. 2014.

[29] Herbert a. Simon. Designing organizations for an information-rich world, 1971.

[30] Ryan Singel. Netflix spilled your brokeback mountain secret, lawsuit claims, Sep 2012.

[31] Boxun Zhang, Gunnar Kreitz, Marcus Isaksson, Javier Ubillos, Guido Urdaneta, Johan a. Pouwelse, and Dick Epema. Understanding user behavior in Spotify. *2013 Proceedings IEEE INFOCOM*, pages 220–224, 2013.

[32] H Zhang. The optimality of naive Bayes. *Aa*, 2004.

[33] Hongbiao Zhao. A dynamic contagion process for modelling contagion risk in finance and insurance. 2012.