An Analytical Survey of Song Recommendation Methods

J. Baker, J. Janzen, N. Moyo, T. Stevens, and U. Waheed (May 2015)

***Abstract*— Predictive algorithms are used in a wide variety of commercial retail applications to improve customer engagement and sales revenue. Oftentimes it is not clear what type of algorithm should be used for a specific use case. This paper examines the effectiveness of three common data analytics algorithms commonly used to implement recommendation engines – Association Rules, Collaborative Filtering, and Naïve Bayes. The researchers used a real-world dataset containing millions of song listener transactions to compare and contrast the predictive capabilities of the algorithms. A comprehensive testing methodology was implemented to generate F1 scores for each of the prediction models. Our analysis suggests that while the Association Rules algorithm offers the best precision, Collaborative Filtering may be a better algorithm in a production setting due to its acceptable precision and lower computational cost.**

***Index Terms*—Association Rules, Collaborative Filtering, Naïve Bayes, Song Recommendation Engine**

I. Introduction

MILLONS of consumers purchase and listen to music on the Internet every year. They purchase MP3s from music distributors such as Apple iTunes or purchase subscriptions to stream their favorite music from companies like Spotify. It’s clear that music distribution is gradually shifting from brick-and-mortar retail stores to the Internet (Sisario, 2014).

Internet music providers have a strong interest in maintaining high customer engagement and customer satisfaction. As customers visit a music provider’s website, they are provided with music recommendations based on current music trends, promotions, and listener behavior. Music providers can increase customer engagement by providing relevant song recommendations that increase customer purchases. Ultimately, better song predictions and recommendations translate into increased revenue for the music provider (Kaufman, 2014).

A key challenge a music provider faces when making song recommendations is which type of recommendation algorithm to use. The provider can select from many different types of algorithms, each with advantages and disadvantages.

Our goal in this project was to analyze three of the most popular data mining algorithms used in recommendation engines in the context of a large dataset containing song listener data. Our paper compares and contrasts these algorithms and offers a final recommendation for a song prediction algorithm.

II. Data Characterization

One of the most important steps to perform before analyzing a data set is to review the characteristics of the data. Understanding the characteristics of the data helps a data scientist identify potential challenges such as frequent data items that may create a majority bias, or rare data items that may disproportionately impact analysis results.

The focus of our analysis for this project was on the Taste Profile Subset, a collection of over 48 million song play transactions – basically, records where an Internet user listened to a song a certain number of times. The Taste Profile dataset provided ample real-world training data for our predictive algorithms.

We also utilized the Million Song Dataset during the project to verify prediction results and understand certain characteristics of the transactional data. The Million Song Dataset contains metadata on 1 million popular songs (pre-2012) including attributes such as the song title, artist name, musical genres, and length. Since the two separate datasets shared the same SongID fields, we were able to join the song metadata with the song listening transactions.

The Taste Profile Subset did not actually contain all 1 million songs from the Million Song Dataset. A little over 380,000 songs, almost 40% of the total songs from the Million Song Dataset were present in listener transactions. Over 110,000 users listened to the most popular song in the Taste Profile Subset. This was the exception to the rule (see Fig. 1) as the number of listeners per song decreased rapidly in the dataset. We expected a small number of songs to be present in a high number of listener transactions, and the data seems to reflect the reality that only a limited number of songs in the marketplace achieve high levels of popularity. In this case, the top 100 songs in the dataset were present in 7% of all the listening transactions.

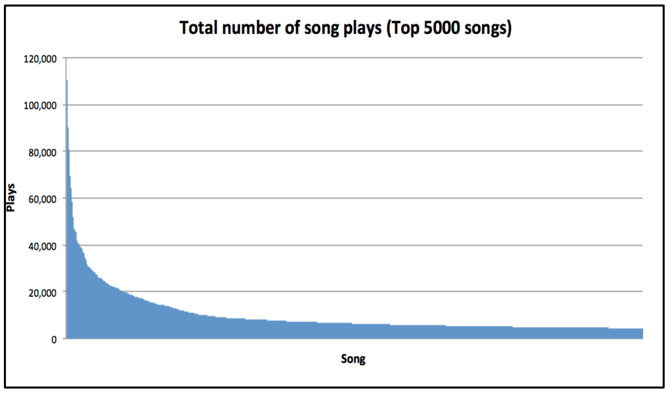

Figure 1. Top number of song plays in Taste Profile Subset

It's important to note that over 22,000 songs were listened to only once in the dataset. These types of rare songs had different influences on each of the predictive algorithms. In some cases, these songs simply were not present in the predictions because their frequency in the dataset was so low.

Every user in the dataset listened to at least 10 songs, which was beneficial from a prediction standpoint. It also meant we didn't have to deal with test cases where a user only listened to a single song. Since we ultimately split user song transactions into two sets for testing purposes, we would not be able to test our prediction for a user that listened to only one song.

We discovered one anomaly while carefully reviewing the data. One of the basic questions we asked was: Which song was listened to by the most users? Let's say that a user listens to Song A ten times and nine users listen to song B once. Technically, Song A was played more often but the second song was selected by more people – and therefore showed up in more transactions. We wanted to find out what some of these frequent songs were to understand how they might impact our predictions.

We wrote a simple query to answer this question and received a head-scratching result (see Table 1). According to the Taste Profile Subset, the song with the largest number of listeners was a song by a group called Harmonia -- a German rock band from the early 1970's. How could a band we had never heard of achieve such fame in our dataset? The answer pointed to a flaw in the data and a bit of quick Internet research confirmed our finding.

Songs don't have universal identifiers in the real world. When we talk about songs, we use song titles, album and artist names, and release dates. The SongId in the Million Song Dataset is an attribute that a group of researchers created to track and index song data (Bertin-Mahieux et al., 2011). The Taste Profile Subset includes listener transactions and song data gathered from a completely different commercial data source (The Echo Nest).

Rank	Plays	SongID	Artist	Song Title
1	110,479	SOFRQTD12A81C233C0	Katy Perry*	Firework*
2	90,476	SOAUWYT12A81C206F1	Bjork	Vespertine Live
3	90,444	SOAXGDH12A8C13F8A1	Florence & The Machine	Now That's What I Call Music
4	84,000	SOBONKR12A58A7A7E0	Dwight Yoakam	If There Was A Way
5	80,656	SOSXLTC12AF72A7F54	Kings of Leon	Only By The Night
6	78,353	SONYKOW12AB01849C9	One Republic	Waking Up
7	69,487	SOEGIYH12A6D4FC0E3	Mozart	Eine kleine Nachtmusik
8	64,229	SOLFXKT12AB017E3E0	Karaoke Monthly Vol. 2	Karaoke
9	63,809	SODJWHY12A8C142CCE	Train	Save Me
10	58,610	SOFLJQZ12A6D4FADA6	Cartola	Nova Bis

Table 1. Top 10 most played songs (* corrected data)

In order to join the Million Song Dataset and the Taste Profile Subset together, the researchers used an algorithm to match song and artist names found in the two datasets. Unfortunately the matching algorithm wasn't perfect, and over 5,700 of the songs were incorrectly matched. The most widely represented song in the dataset wasn't really from an obscure German band, but from the immensely popular U.S. singer Katy Perry.

The data matching flaw highlights the fact that large datasets are never perfect, especially when the dataset represents a combination of data from disparate sources. We would not have noticed this anomaly without asking some basic questions about the data and trying to make sense of the results. Perhaps the greatest tool of a data scientist is common sense. In this case, the fact that some of the SongIds were matched to incorrect metadata values did not diminish the quality of our results. If we were going to use this dataset in a commercial pursuit, we would need to correct these metadata mismatches before launching a product.

III. Tools Selection

Our team used a variety of tools for this analysis project including: SAP Hana, Apache Hadoop, Apache Hive, SQLite and Microsoft Excel. We used HANA and Hadoop/Hive to execute the data analysis algorithms. We ran algorithms on HANA using the built-in Predictive Analysis Library, whereas on Hadoop we leveraged Hive SQL (HQL) statements. Microsoft Excel was used during the testing and analysis phase of the project.

We divided the project work into four distinct stages and the team members assigned to each stage selected a tool that met their needs. The project stages included:

- Data acquisition and pre-processing
- Data algorithm execution
- Algorithm testing
- Data analysis

The data pre-processing work for the project was fairly straightforward. We didn't use the full Million Song Dataset for this project, but elected to use a subset of the song metadata stored in an SQLite database. The metadata contained a number of important fields including the songId, song title, and artist name. We converted the SQLite database into a TSV file for import into HANA and Hadoop.

A few of the advantages of using Hadoop/Hive for processing are that it is widely used in many Big Data analysis projects, it supports an accessible SQL-like language, and it is highly-scalable (Henschen, 2015). One of our team members has extensive experience working with Hadoop at a large retailer, and his expertise allowed us to quickly bootstrap a data-mining environment.

One key disadvantage of Hadoop/Hive versus SAP HANA is that Hadoop does not leverage in-memory processing and therefore can lack the speed of HANA. Hadoop makes up for this disadvantage by supporting the addition of computing nodes to a Hadoop cluster. Another disadvantage of Hadoop/Hive is that there are no default analytical functions to process data. We had to manually create the data analysis algorithms on Hadoop and process data in a batch-like fashion.

We used a 30-node Hadoop cluster (XL size) to process datasets containing 137 million rows and 2.2 billion rows of data. It's likely that we could have used a smaller cluster to contain costs, but in this case the processing was relatively quick and cost wasn't a major contraint.

Microsoft Excel was used to analyze the result sets, calculate statistics, create visualizations for a comparison of the algorithms' performances and facilitate a final recommendation.

Using Excel in a data-mining project has a number of advantages. It is widely available on most workstations in professional and academic environments. We didn't need to expend cloud-computing resources to access Excel. The software has a large library of powerful and easy to use functions – including nested functions.

We used Excel to create quick data visualizations using a large set of commonly used visualization templates. Finally, we could also quickly repurpose code, pasting different datasets into a template to quickly generate uniform output.

One of the major weaknesses of Excel was processing large datasets like the ones we used in our project. The i5 3.GHz processors on our local machines were bottlenecks when trying to execute large numbers of calculations.

Disabling the automatic updating of calculations mitigated the resource limitations of our local machines, and allowed transactions to be manually committed once the calculations for a new data series had been set up. This was a useful step to take since a single update to a threshold could take upwards of 20-minutes to process.

IV. Algorithms

We selected three common data mining algorithms to generate predictions for our song listening dataset: Association Rules (AR), Naïve Bayes (NB), and Collaborative Filtering (CF). Each algorithm utilizes a different methodology to produce predictions. For example, Association Rules is an eager algorithm that generates a set of rules based on the pairing of songs in listening transactions, whereas Naïve Bayes and Collaborative Filtering represent lazy algorithms. Association Rules provides a confidence score and Naïve Bayes provides a probability for the predictions. Collaborative Filtering simply offers a prediction ranking.

Each of the algorithms implemented in the project exhibited certain advantages and disadvantages. Let's look at how we implemented these algorithms and how algorithm input parameters impacted our prediction models.

A. Association Rules

We implemented the Association Rules data-mining algorithm on our song dataset using the latest version of SAP HANA Developer Edition running on an Amazon EC2 instance. HANA provides support for the AR Apriori algorithm as part of its Predictive Analysis Library (PAL).

The Taste Profile Subset dataset is perfectly suited for the AR algorithm since it represents a set of transactions consisting of user IDs and song IDs (see Table 2). Once the dataset was imported into HANA, no data pre-processing was necessary before executing the algorithm. The Apriori procedure requires an input table with an ID and an ATTRIBUTE. We created a simple SQL view mapping the UID and SONGID fields from the dataset to these respective input fields.

We did not take into account the "Play count" field during the implementation of the Association Rules algorithm. Each record represented a single item in a transaction, essentially treating all items uniformly. An interesting future path of inquiry might involve adding a weighting factor to items based on the recorded play count.

The HANA Apriori algorithm accepts a number of required and optional control parameters. The two key control parameters are the minimum support and minimum confidence settings. The minimum support provides a way to filter out extremely rare songs that offer almost no predictive value. For example, if only one user listened to a song it is difficult to make any sort of reasonable prediction based on that single transaction. However, if 10 users listen to a song then the accuracy of the prediction is likely to be higher. We set a minimum support level representing 10 users in the dataset.

UserID	**SongID**	**Play count**
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAPDEY12A81C210A9	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBFNSP12AF72A0E22	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBFOVM12A58A7D494	1

Table 2: Taste Profile Subset records

The minimum confidence level is also important. However, the exact confidence values wasn't as important during the rule generation since we were able to change the confidence level threshold in our rule set during the testing process. Our goal during the rule set generation was to set the confidence level low enough to capture a large number of rules representing a range of confidence levels. We set a minimum confidence level of 20%.

The selection of the MAXITEMLENGTH control parameter was problematic on the HANA Developer Edition instance we used for the project. Optimally, the maximum item length would represent a meaningful number of items since Association Rules with a larger number of items in the antecedent should provide better predictions. However, the Taste Profile Subset contains so many large frequent item sets that anytime the maximum item length parameter was set to a value greater than 3, the HANA instance would eventually run out of memory.

Fortunately, our prediction model did not need to use Association Rules with more than two items. We were able to take the generated rules and rank them by confidence in order to provide a set of song recommendations. In many ways this actually made the recommendation process much easier. A future iteration of this project could involve testing the prediction capability of the Apriori algorithm when generating longer item sets.

One potential flaw of our implementation is that we did not account for cross item sets during our analysis. Relatively rare songs in the dataset could have been associated with very frequent songs at a high level of confidence. This type of cross support could have adversely impacted prediction performance. The support ratio measurement could have been used to filter out cross support associations (Tan, 2005).

The HANA Apriori algorithm took approximately 1 hour and 45 minutes to process the 48 million transactions in the Taste Profile Subset. The algorithm generated over 1.5 million rules containing two items (see Table 3).

PRERULE	POSTRULE	SUPPORT	CONFIDENCE	LIFT
SOXKSAJ12A8C14588D	SOGDNKT12A8C1447DA	0.00016972132347314577	0.46630727762	1037.80655374
SOZZWDF12A8C14144C	SOKPDCL12A8C13B5E7	0.00016972132347314577	0.64552238805	1076.91094853
SOFLQTC12A67021CCA	SOCHRFO12AF729B18C	0.00015892979423496887	0.41968911917	1490.58074416

Table 3: Apriori algorithm generated rules.

One of the key advantages of the Association Rule algorithm is that it is able to provide explanations for song recommendations. If the algorithm recommends song S^3, it can also explain that songs S^1 and S^2 were the reasons for the recommendation. It can provide a level of confidence to support the prediction. The AR algorithm is also adept at recommending less popular songs – as long as those songs meet a minimum support level (Forster, 2013). In a sense, Association Rules offer a more personalized recommendation to song listeners.

While the AR algorithm offers many advantages in the context of a song recommendation model, it also has a number of disadvantages. It's a relatively expensive algorithm compared to Naïve Bayes and Collaborative Filtering. Music providers implementing the algorithm would likely encounter scalability issues as the number of music subscribers and transactions grow. Also, it doesn't have the ability to offer predictions for new or rare songs since those songs likely will not meet minimum support thresholds.

B. *Collaborative Filtering*

The Collaborative Filtering algorithm is a crowd-based recommender method that finds users with similar listening behaviors and ranks their song choices. Collaborative Filtering (CF) was the simplest to implement and least expensive to run of all three algorithms used in our project. We only had to execute two SQL statements to create a dataset of ranked song recommendations (see Figure 2). The algorithm was able to process over 48 millions rows of data in approximately 30 minutes.

The logic used to design our algorithm was based on User-to-User Collaborative Filtering, which involves setting a Boolean value based on whether a user listens to a song or not --regardless of the number of song plays.

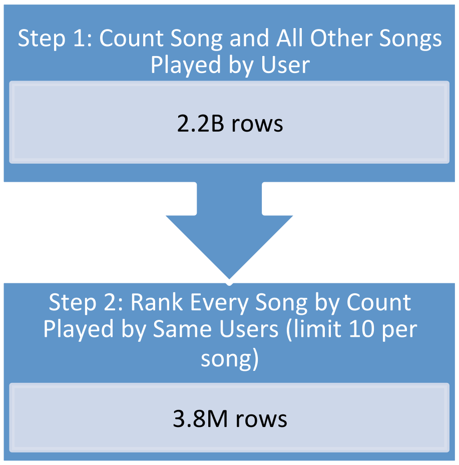
Step 1: Count Song and All Other Songs Played by User
2.2B rows
Step 2: Rank Every Song by Count Played by Same Users (limit 10 per song)
3.8M rows

Figure 2: Collaborative Filter SQL steps.

If a new user were to listen to song D (see Table 4), we would identify the other users who had also listened to song D, which are U{2,3}. Then, by looking at the other songs these users listened to, we sum up the count of each song and the song with the highest total ranking would be the song recommendation for the new user.

Song	A	B	C	D
User 1	1	1	1	0
User 2	0	1	0	1
User 3	1	1	0	1

Table 4. User play history.

Based on the user song transactions, the Collaborative Filtering algorithm generates a set of song recommendations. The table (see Figure 2) shows that song B would be ranked #1, followed by song A. Song C would not be recommended because the users U{2,3} did not listen to that song.

Recommended Song	User count	Rank
B	2	1
A	1	2

Table 5: Recommendation for Song D

Collaborative Filtering is a fast and efficient predictive algorithm that makes predictions based on the song rankings of users that listened to similar songs. The simplicity and general accuracy of the algorithm are its core strengths (Lee, 2011). The algorithm is scalable and does not exhibit some of the inherent performance limitations of an algorithm such as Association Rules. Since the algorithm is essentially just maintaining a set of item counts and relationships, it is easily adaptable to in-memory database implementations.

Besides speed of processing and ease of implementation, another benefit of CF is a deep recommendation set (Reichlin, 2008). There really is no limit to the number of songs recommended as long as the song used as the basis for a prediction was listened to by at least one user in the dataset.

Collaborative Filtering isn't a perfect algorithm. Like Association Rules, it has trouble making recommendations for new songs. It exhibits a "cold start" problem where it needs to have enough listeners in the system before it can find song matches and make predictions. If a new song is added to the song catalog, it will have no user listening history and therefore will not be recommended. Also, a certain amount of ramp up time is associated with new songs to increase song ranking in prediction results (Mackey, 2009).

The algorithm suffers from a type of popularity bias where it struggles to recommend items to listeners with unique music tastes. It tends to recommend the most popular songs. Finally, when the Collaborative Filtering algorithm makes a prediction it simply provides a song ranking with no additional reasoning. In other words, it provides a song prediction but not the reasons why it made the prediction.

C. Naïve Bayes

Naïve Bayes was slightly more complex to implement than CF, and required four steps of SQL statements to produce the recommendation dataset. The total processing time of the algorithm generating probabilities for 48 millions row of data was approximately 60 minutes (see Table 6).

SongId	Pred_SongId	Plays	Adjust_p	Rank
SOAANAX12AB017E7F3	SOPVBQE12AB018613D	9	0.7500066221332294	1
SOAANAX12AB017E7F3	SOUIZHA12AB0181516	6	0.750004414742493	2
SOAANAX12AB017E7F3	SOVSNUO12AB01814FA	4	0.666669282805233	3

Table 6: Naive Bayes probability results

The Naïve Bayes algorithm basically makes song predictions by examining the prior probability of songs in the dataset and comparing those probabilities to a set of target songs used in the prediction. For example, let's say that based on user play history (see Table 4) we want to calculate the probability of song A played with driver song D. The NB algorithm performs a series of probability calculations to determine a result:

Probability both songs played with driver song D = (count U played song A & D)/(count U listened song D) = 0.5

Probability both songs not played = (count total U – count U played song A & D)/(count total U) = 0.67

Adjusted probability song A is played with driver song D = 0.5/(0.5+0.67) = 0.431

The adjusted probability is calculated for every song in the dataset creating a set of songs and predicted songs with an associated probability score. The recommended songs are then sorted by probability in descending order during the recommendation process.

Creating the NB results set involved running four HQL statements on a Hadoop cluster (see Figure 8).

One of the key design inputs we implemented in this algorithm was to only calculate the probabilities for songs with a minimum of 10% plays of the driver song. When we first ran the algorithm without this constraint, the probability results contained a number of unusual song predictions with high probabilities. Filtering out some of the infrequent songs improved prediction results.

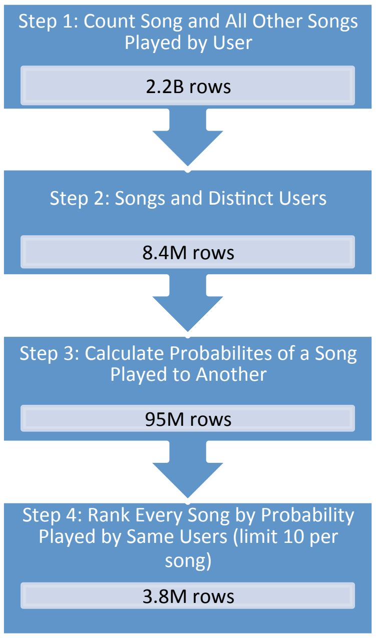
Step 1: Count Song and All Other Songs Played by User
2.2B rows
Step 2: Songs and Distinct Users
8.4M rows
Step 3: Calculate Probabilites of a Song Played to Another
95M rows
Step 4: Rank Every Song by Probability Played by Same Users (limit 10 per song)
3.8M rows

Figure 3: Naive Bayes SQL steps.

Naïve Bayes has some of the same advantages and disadvantages of the Collaborative Filtering algorithm. Like Collaborative Filtering, Naïve Bayes is relatively simple to implement and produces reasonably accurate predictions.

One of the subtle complexities and disadvantages of the algorithm is that certain types of probability estimates require corrections (Lavrenko, 2015). For example, if the frequency of a class is zero in a probability calculation, the zero calculation essentially has veto power over the probability result. This regrettable situation is remedied by the use of a Laplacian correction during the calculation process. Once Naïve Bayes provides a prediction, there is no way to know the rules or reasoning behind the prediction. The algorithm simply provides a probability score.

D. *Visual algorithm comparison*

One of the simplest ways to test whether or not our algorithms were making sensible predictions was to select a driver song and compare the prediction results of the three algorithms. Our expectation was that the algorithms would provide similar results, but perhaps with different rankings.

In Tables 7, 8 and 9 below, the algorithms made predictions based on a driver song from the popular band *U2* called *Endless Deep*. The prediction results of the three algorithms compare favorably. Recall that CF creates its rank based on user count, NB uses the adjusted probability, and AR uses confidence. The three algorithms all share a set of five song recommendations in common, although these recommendations are offered in differing rankings.

Title	Users	Rank	NB rank	AR rank
Dancing Barefoot	175	1	1	2
Hold Me_ Thrill Me…	175	2	10	1
Love Comes Tum…	164	3	6	3
Walk To The Water	144	4	2	4
A Day Without Me	84	5	3	5
Window In The Skies	82	6	-	6
I Still Haven't Found…	78	7	-	7
Bad	76	8	8	-
Vertigo	57	9	-	-
Mysterious Ways	55	10	9	-

Table 7: Collaborative Filtering results (U2 - Endless Deep)

Title	Adjust_p	Rank	CF rank	AR rank
Dancing Barefoot	0.22	1	1	2
Walk To The Water	0.20	2	4	4
A Day Without Me	0.07	3	5	5
Always	0.05	4	-	-
Another Time Anot…	0.05	5	-	-
Love Comes Tum…	0.04	6	3	3
Fast Cars	0.04	7	-	-
Bad	0.04	8	8	-
Mysterious Ways	0.03	9	10	-
Hold Me_ Thrill Me…	0.03	10	2	1

Table 8: Naive Bayes results (U2 - Endless Deep)

Title	Confidence	Rank	CF rank	NB rank
Hold Me_ Thrill Me…	0.45	1	2	10
Dancing Barefoot	0.45	2	1	1
Love Comes Tum…	0.42	3	3	6
Walk To The Water	0.37	4	4	2
A Day Without Me	0.22	5	5	3
Window In The Skies	0.21	6	6	-
I Still Haven't Found…	0.20	7	7	-

Table 9: Association Rules Results (U2 - Endless Deep)

V. Testing Methodology

The initial step in our testing process involved creating two sets of data: a training set and a testing set. The training dataset was used to build the prediction models, and the testing dataset was used to calculate the quality scores. For this project, a random sample of 1,000 records was selected from the larger dataset of 48 million records. These 1,000 records comprised the testing data and were set-aside during the model training. The remaining records comprised the training data that the different algorithms ingested to build their prediction models.

Once the models were built from the training dataset, the testing data set was used to generate predictions. For each set of user transitions in the testing set, the playlist of songs was split in half. The first half of the playlist was input into each model with the goal of predicting the second half. Each song was input, one at a time, as a song driver into one of the predictive models. The set of songs returned from the model created the list of recommended songs.

Example:

A user has the playlist {A,B,C,D,E,F}
Split the playlist in half using {A,B,C} to test the model to predict {D,E,F}
Entering {A} into the model returns {D,G,H,I}
Entering {B} into the model returns {D,E,G,I}
Entering {C} into the model returns {E,H,I}

Once the list of recommended songs was generated, it was compared to the second half of the original playlist -- the expected set. The result of this comparison was used to create a confusion matrix (see Table 10). Every song from the recommended set that was also in the expected set was denoted as a true positive (i.e., the song was recommended correctly).

Every song that was in the recommended set, but that was not in the expected set was denoted as false positive. In other words, the song was recommended, but should not have been. Every song that was not in the recommended set, but that was in the expected set was considered a false negative. The song was not recommended, but should have been.

Those songs that were not in the recommended set and not in the expected set were the true negatives. The song was not recommended and should not have been recommended. Note, the true negative measurement was not used in the analysis of the models and was therefore not presented in this paper.

Continued example:

Aggregate the results to the set {D,D,E,E,G,G,H,H,I,I,I}
Compare the expected set {D,E,F} to the recommended set {D,D,E,E,G,G,H,H,I,I,I}
*True Positives = {**D**,**D**,**E**,**E**,G,G,H,H,I,I,I} = 4*
*False Positives = {D,D,E,E,**G**,**G**,**H**,**H**,**I**,**I**,**I**} = 7*
*False Negatives = {D,E,**F**} = 1*

	Expected (Yes)	Expected (No)
Recommended (Yes)	4	7
Recommended (No)	1	--

Table 10: Confusion Matrix for example

With the creation of a confusion matrix, it was possible to compute a number of metrics such as: accuracy, precision, misclassification rate, specificity, prevalence, recall and F1. The comparison of the three models used in this project focused on recall, precision and F1. An additional measurement, average number of recommendations per song, was also computed. This measure provided a uniform basis of comparison when analyzing the performance of the three algorithms at varying thresholds.

Sample calculations:

Recall (R) = True Positive / (True Positive + False Negative) = 4 / (4 + 1) = 0.80

Precision (P) = True Positive / (True Positive + False Positive) = 4 / (4 + 7) = 0.37

*F1 = 2 * (P * R) / (P + R) = 2 * (0.37 * 0.80) / (0.37 + 0.80) = 0.51*

Average Recommendations per song = (True Positive + False Positive) / Count({A,B,C} = (4 + 7) / 3 = 3.67

VI. Analysis and Results

A. *Naïve Bayes*

The testing methodology was applied to each model beginning with the Naïve Bayes classifier. As in the example above, each song in the first half of the playlist was entered as an input into the model generating a list of recommended songs which, when aggregated, totaled 213,760 recommendations for the testing set. Each record contained a User ID, Song ID, Recommended Song ID, probability of the recommendation and a binary indicator denoting whether or not the recommended song was in the second half of the playlist for that UserID (see Fig 4). The addition of the Probability attribute in the dataset serves as a threshold to further segment the results. We used confidence scores and user ranking in a similar fashion for Association Rules and Collaborative filtering respectively.

User ID	Song ID	Recommended Song ID	Probability	Actually Played

Figure 4: Naive Bayes data fields

From the algorithm results set, it was possible to calculate metrics for evalutating the performance of the model. The number of True Positives equaled the count of records with a 1 in the Actually_Played field and the number of False Positives equaled the count of those records where that attribute equaled 0.

The number of False Negatives is the count of those songs in the expected set that were not recommended. However, the number of songs in the expected set was not explicit from the results set and further data processing was required.

The user's playlist was initially halved to create the two sets of songs: those that were entered as parameters and those that were expected as output. These two sets were of equal size. By summing the distinct User ID – Song ID combinations in the results sets, it was possible to determine the number of songs that were entered as a parameter into the model and thus determine the number of songs in the expected set. The size of each set was consistent across all three models and only needed to be calculated once. A total of 22,620 songs were entered into each model, and the same number was expected in the results set.

In addition to deriving the total number of expected songs, it was necessary to calculate the number of songs that were recommended. Simply using the count of True Positives was not sufficient as those numbers included songs that were recommended more than one time. Selecting the distinct number of User ID – Recommended Song ID combinations where Actually_Played was equal to 1, eliminated these more frequent recommendations. The number of False Negatives equaled the number of expected songs minus the number of distinct actual plays and the confusion matrix was complete.

As stated, the addition of the probability attribute allowed us to further segment the results set. By selecting only those records where the probability was greater than or equal to a minimum threshold, multiple confusion matrices were created that were used to generate comparison metrics. For example, using the Naïve Bayes model 21 probability thresholds were created ranging from 0% to 100%. Recall, Precision, F1 and Average Number of Recommendations were plotted against each other and an optimum threshold for the model was identified (see Fig 5 & 6).

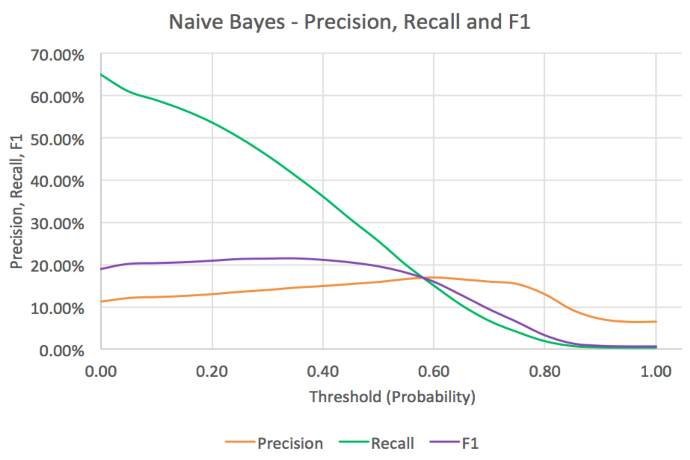

Figure 5: Naive Bayes quality

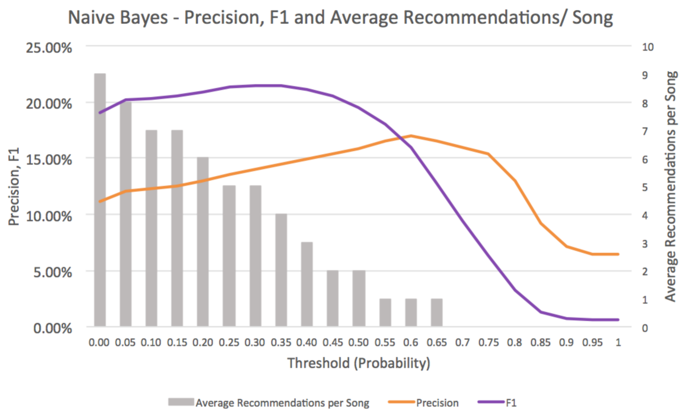

Figure 6: Naive Bayes avg recommendations/song quality

These charts show that as the probability threshold increases, the Recall of the Naïve Bayes classifier behaves as expected – basically recommending fewer songs than are expected to be recommended. The model's Precision, the measure of how many recommendations are correct of the total recommendations, also performs as expected.

As the threshold increases, the model accurately recommends more songs with fewer recommendations until the threshold reaches approximately 60%. This threshold represents the maximum Precision, from which point the model is not able to accurately recommend songs at the same rate of recommendations.

The F1 score utilizes both Recall and Precision for a given threshold and assists in determining the point of optimum performance across the two measurements. The chart suggests that the best F1 scores occur when the threshold ranges from 10% to 45% with a peak at 35%. The project objective was to return as many songs of interest as possible, therefore the average number of songs recommended must also be considered. Focusing on the threshold range with the highest F1 scores, an expected gradual decline in average recommendations was seen ranging from 7 recommendations at 10% to 2 recommendations at 35%.

Due to a high F1 score amid a large number of recommendations, a threshold of 30% probability provides the best performance in meeting the project objective when using a Naïve Bayes classifier.

B. Association Rules

The same set of songs that were input as attributes into the Naïve Bayes classifier were next input into a model that uses Association Rules to determine its recommendations. This model also returned a list of recommendations for each song parameter in a similar format as the Naïve Bayes model (see Fig 7). The only difference between the two data structures was that rather than the probability of the recommendation, this model returned the confidence of each recommendation as determined from an Association Rule.

User ID	Song ID	Recommended Song ID	Confidence	Actually Played

Figure 7: AR model data fields

In the same manner as the Naïve Bayes model, it was possible to calculate True Positive, False Positive and False Negative scores from the results set of 128,191 recommendations. Setting a minimum threshold ranging from a confidence greater than or equal to 0.20 to a confidence greater than or equal to 0.90 further refined these recommendations, and a confusion matrix was created at each threshold. The metrics -- Recall, Precision, F1 and the Average Number of Recommendations -- were then plotted for ease of comparison with the other prediction models (see Fig 8 & 9).

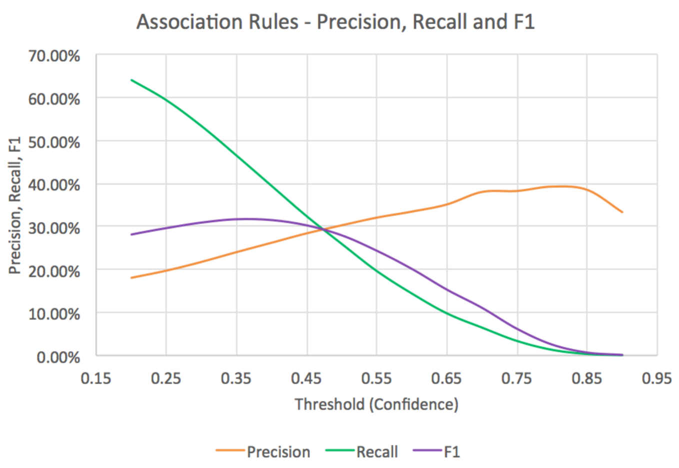

Figure 8: AR quality scores

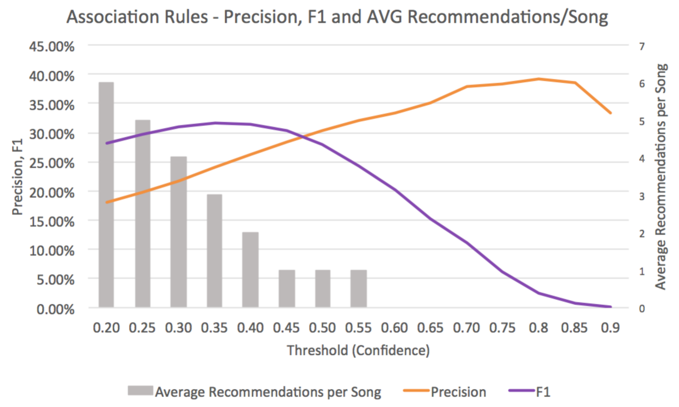

Figure 9: AR average recommendations/song quality

The first observation regarding the Recall and Precision graphs is that their slopes, or rate of change between thresholds, stay relatively consistent before flattening out at the higher confidence thresholds. This consistency in the model at different thresholds is useful as it may allow for better quality prediction at lower thresholds. For example, if a new threshold was set at 0.10 we may predict a Recall of 74% and a Precision of 16%. Contrast this with a chart where the slope varies greatly from one threshold to another and it becomes much more difficult to predict what will occur at the new threshold.

A second observation is that the slopes were relatively steep. The maximum Recall is 65% at a confidence of 0.20 and drops to 1% at a confidence of 0.80. While not as significant of a change, the Precision increases by 20% over the same range of thresholds. The net takeaway from this analysis is that both Recall and Precision are highly dependent on the threshold.

The Recall and Precision charts, however, do not tell the entire story. Again, one must observe the average number of recommendations at a given threshold. At a confidence equal to 0.60, the average number of recommendations is less than one. If the threshold is 0.60 or greater, users are unlikely to receive recommendations and this would not satisfy the project objective. Furthermore the confidence must be lowered to 0.25 in order to return 5 recommendations on average. The F1 score for this model peaks at a confidence equal to 0.35 where an average of 3 recommendations are made. Therefore, in order to meet the objective of returning a meaningful number of recommendations, the F1 score will not be optimal.

A threshold where confidence is equal to 0.25 best satisfies the project objective when using an Association Rules model.

C. Collaborative Filtering

The set of songs that were previously entered into the both the Naïve Bayes and Association Rules models was entered into the Collaborative Filtering model to produce a third set of results. Again, the results set was similar to the first two with the only difference being the measurement in which to create a threshold (see Fig 10). Collaborative Filtering recommends songs based on the frequency of the recommended song being played. The song with the highest frequency is the first recommendation; the song with the second highest frequency is the second recommendation and so on. Each recommendation's ranking was included in the results set.

User ID	Song ID	Recommended Song ID	Rank	Actually Played

Figure 10: Collaborative filtering model data fields

The threshold for this results set was driven by the rank of the recommendations. The lowest threshold included all recommendations that were ranked 10 or higher, and the highest threshold included only recommendations where the rank equaled 1. Using the same formulas as the previous two models, ten confusion matrices were created and the metrics plotted (see Fig 11 & 12).

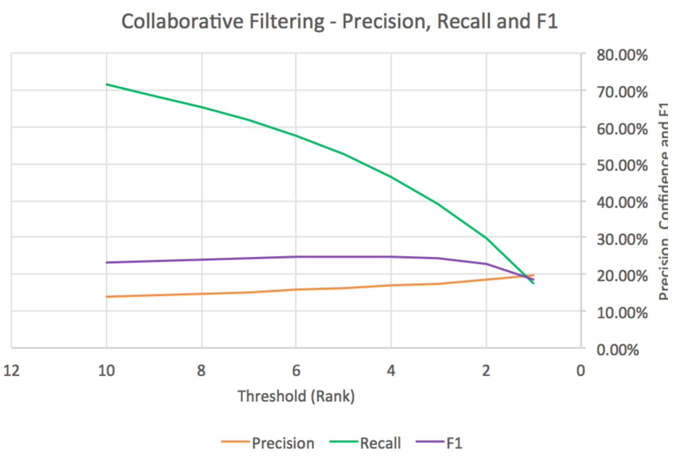

Figure 11: Collaborative filtering quality scores

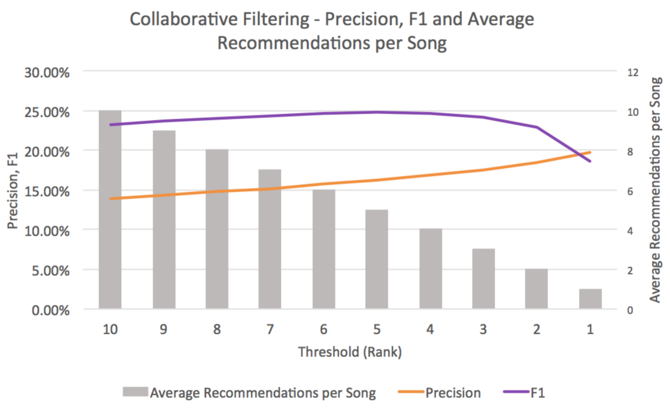

Figure 12: Collaborative filtering average recommendations/song quality

Of primary interest in the charts is that the F1 score is relatively flat across the thresholds suggesting that the performance of the model is not strongly coupled to the threshold. Investigation of the two components that comprise the F1 score illustrate why this is the case. The Precision illustrates that the model recommends the correct songs at a rate proportionate to the number of recommendations, thus the slope of this line is also relatively flat across the different thresholds.

As the model performs equally well across the thresholds, it is much easier to select a threshold that maximizes the project objective. Setting the threshold to 5 returns a large number of recommendations with minimal declines in F1 and Precision.

D. Comparing the models

In addition to analyzing the three models individually, it was necessary to do a cross comparison in order to select the model that best satisfied the project objective: recommend a large number of songs of interest to keep a listener engaged. In other words, return a high number of average recommendations per song while maintaining good precision.

We noted in our analysis earlier that for Naïve Bayes and Association Rules, the portions of the precision curves that indicated the highest precision also yielded 0 or very few average recommendations per song. This implies that we are unable to use these algorithms at higher thresholds because their real-world performance would not be satisfactory (i.e. 0 or 1 song recommended on average for each song fed into the algorithm).

Collaborative Filtering showed a positive, linear relationship between the rank-threshold and the average number of songs recommended for the range of thresholds tested. This means that while it too could not recommend a large number of songs at the highest thresholds, the number of recommendations it provided could be more readily scaled. While confidence and probability for Association Rules and Naïve Bayes have a lower bound of 0, the rank threshold for Collaborative Filtering could have been extended past 10 to yield more average recommendations per song.

Another way to look at this issue is found in the following data from our testing, which shows the number of driver songs (not User-Song combinations) that actually had at least one recommendation:

Association Rules:	12,754
Naïve Bayes:	15,341
Collaborative Filtering:	15,392

As we might expect, Association Rules will not generate a recommendation for songs for which it has no rules, which places a limit on its utility.

While these aggregated figures relate specifically to our test data, we feel that taking the average recommendations per song for each song in our test set allows us to compute a measure that is somewhat more generalizable[1] for the performance of these algorithms on our complete data set.

With regard to precision, Association Rules is able to attain the highest precision of all the algorithms, but its precision also shows the steepest decline as the threshold increases.

[1] According to the Law of Large Numbers, as the sample size increases, the sampling distribution approaches the true mean of the population and the variance of the sampling distribution becomes smaller. Our average recommendation per song calculations are based on 22,620 cases fed to each algorithm and we feel that this will allow us a meaningful estimate of the average recommendations per song that each algorithm is able to generate.

The important observation about Collaborative Filtering is that its precision shows a gradual decline relative to Association Rules' steeper slope and, at the limit of the thresholds in our test-results, it was able to generate more average recommendations per song without a precipitous decline in precision.

Naïve Bayes' precision consistently tracks below that of the other two algorithms and while it was able to generate more average recommendations per song than Association Rules, it does not add any value when compared to Collaborative Filtering on these two measures.

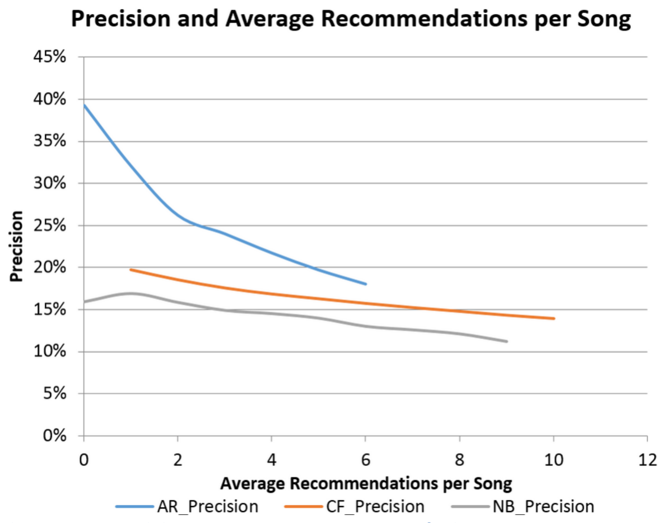

Figure 13: Precision vs Average recommendations per song

A comparison of Precision and Average Recommendations per Song (see Fig. 13) provides a succinct way to compare the performance of the three algorithms. The chart shows that while Association Rules and Naïve Bayes algorithms are unable to make noticeable recommendations at their highest precision, the precision curves of all three algorithms overlap from the 1 through 6 range -- or average recommendations per song. Throughout that range, it is evident that Association Rules offers the highest precision and Naïve Bayes returns the lowest.

E. Implementation and other considerations

Our examination of a predictive model must also include external considerations such as implementation, processing speed and biases. The Collaborative Filtering algorithm was easy to implement, fast to train and had minimal storage requirements. Collaborative Filtering is a lazy classifier; it will not make recommendations until the user asks for them. This allows the model to constantly learn and make more adaptable recommendations.

The Association Rules algorithm represents an eager classifier; it needs to create its recommendations prior to the user asking for them. The Association Rules model needs to mine rules based on features such as support, confidence and lift and the inputs to these features may change over time. The model also requires a significant amount of time and memory to create the rules.

The Collaborative Filtering model requires that neighbors are located prior to the user asking for recommendations. When recommendations are needed, the model is then able to filter the neighboring records in order to make accurate recommendations. While not as feature dependent as the Association Rules model, Collaborative Filtering still requires processing time and memory to locate neighboring records. The Collaborative Filtering model has the added difficulty of locating neighbors when a user has only listened to a few songs, an issue known as "cold start". As the user listens to more songs, additional data points are created and the user's neighbors can be refined to those who are most similar.

All three models have difficulty recommending songs that are played infrequently. The Naïve Bayes model will give that recommendation a low probability and the Association Rules model may never generate a rule for that song if it doesn't meet the support requirements. The Collaborative Filtering model recommends songs based on frequency and therefore the recommendation will be at the bottom of the list.

VII. Recommendation

After analyzing the three models and weighing their performance benefits and deficiencies, the model that best satisfied the project objective was Association Rules. While not effective at higher thresholds due to a low average number of songs recommended, the model outperformed the others when the thresholds were adjusted so that each model recommended an average of five or six songs. In addition to a strong performance among the various metrics, this model was also able to provide an explanation as to why a song was recommended.

The Collaborative Filtering model's strong performance across the various thresholds suggested that the model was the most flexible. However, the objective was to predict a set of songs with the most accurate prediction results. When we used an average of 5 songs for model comparison, Collaborative Filtering did not perform as well at that threshold as Association Rules. Factors such as cold start, popularity bias and the lack of an explicit explanation for the recommendation were also considered in deciding that the Collaborative Filtering model was not the best in satisfying the project objective. However, it is worth noting that Collaborative Filtering may be the algorithm best suited for a production environment where more than 5 song recommendations are required at lower computational cost.

The Naïve Bayes model had the second lowest computational requirements of the three models in our test environment. In a constrained computational environment, this model will provide reasonably precise recommendations. The Naïve Bayes model did not perform as well as the other two models at any threshold. Therefore, the model did not best satisfy the project objective.

VIII. Conclusion

While these algorithms are powerful tools to find the implicit relationship between songs, one interesting fact is that, even now, large music streaming services like Google and Pandora still use human beings to curate their playlists. At Pandora, "a team of roughly 30 people, most of them active musicians, analyzes 10,000 songs a month, tagging 200 to 400 traits for each song…" (Samuelson, 2015). Despite the power of our tools, we are reminded of the adage that *the tool will always give results; it is human beings who have to make the important decisions*.

References

[1] Forster, A. (2013, September 12). The SAP HANA DJ - Have some Fun with a Predictive HANA Application. Retrieved May 11, 2015, from http://scn.sap.com/docs/DOC-46289

[2] Henschen D. 16 Top Big Data Analytics Platforms. InformationWeek website, http://www.informationweek.com/big-data/big-data-analytics/16-top-big-data-analytics-platforms/d/d-id/1113609. January 30, 2014. Accessed May 10, 2015.

[3] Kaufman, Jaime C., "A Hybrid Approach to Music Recommendation: Exploiting Collaborative Music Tags and Acoustic Features" (2014). *UNF Theses and Dissertations*. Paper 540.

[4] Lavrenko, V and Goddard, N. *Introductory Applied Machine Learning (Naives Bayes)* - Retrieved from http://www.inf.ed.ac.uk/teaching/courses/iaml/slides/naive-2x2.pdf Accessed 2015, May 9.

[5] Lee, D (2011, February 16). *Collaborative Filtering Based Recommendations*. Retrieved from http://www.pitt.edu/~peterb/2480-012/CF-2011.pdf. Accessed 2015, May 9.

[6] Mackey, L (2009, October 18). *Collaborative Filtering (Practical Machine Learning)*. Retrieved from http://web.stanford.edu/~lmackey/papers/cf_slides-pml09.pdf. Accessed : 2015, May 8.

[7] Reichlin, F. (2008). History-Based Collaborative Filtering for Music Recommendation. Retrieved April 23, 2015, from http://disco.ethz.ch/theses/fs08/report_reichlin.pdf

[8] Samuelson, T. (2015, April 15). For streaming sites, playlists still need human touch. Retrieved May 8, 2015, from http://www.marketplace.org/topics/business/streaming-sites-playlists-still-need-human-touch?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed: APM_Marketpl ace (APM: Marketplace)

[9] Sisario, B. (2014, September 25). U.S. Music Sales Drop 5%, as Habits Shift Online. Retrieved May 2, 2015, from http://www.nytimes.com/2014/09/26/business/media/music-sales-drop-5-as-habits-shift-online.html?_r=0

[10] Tan, P., & Steinbach, M. (2005). Introduction to data mining. Boston: Pearson Addison Wesley.

[11] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

Project Files

Project Handbook and downloadable links to datasets, code scripts, and analysis worksheets located at:

https://github.com/jjanzen/recommender