Million Song Dataset

Year Prediction, Clustering, and Natural Language Processing

Ryan Elson

Abstract

Using a subset of the Million Song Dataset (MSD), I try and predict the year a song was released based on features like the average and covariance timbre values for each song, song duration and loudness, and others. Models built include linear regression, ridge regression, LASSO, random forest, boosting, and others to predict the response, year.

I also used k-means to cluster the songs to understand if, based on the chosen features, the songs are naturally grouped into time periods.

Finally, I use natural language processing (NLP), specifically term frequency-inverse document frequency (TF-IDF) to find the words that were most indicative of a song from a particular decade.

Introduction

The problem of predicting song year is an important one for marketing and music companies, so it is beneficial to have models that can do this accurately. One reason the year is important is that a consumer of a certain age may like songs they grew up with and be more interested in purchasing or listening to songs from that time period. It makes sense to advertise or play these songs if you believe the consumer may want to purchase or listen to them. This type of advertising and playlist building can improve a company's profits, either directly through sales or by driving more users to a music platform which can lead to new subscribers or improved advertising opportunities.

Year is also important since it can also be a proxy for music type. If music details are not readily available, but other songs a consumer listens to are from a genre that's strongly associated with a year then it may make sense to advertise or play songs from said year.

Challenges of building models to predict songs' years are multiple. One problem is the amount of data means that a lot of processing power is required. Using song elements like pitch and timbre leads to a dataset with lots of features. In addition, models should be trained with lots of observations. Some ways to address these issues are to use smaller datasets, cloud-computing, or to use principal component analysis to reduce the dimension being worked in.

Another challenge is that music doesn't often change in a significant way from one year to the next which makes predictions more difficult. One option to address this is to use ensemble methods to improve predictive power.

Throughout the rest of this report, I discuss the dataset and perform an exploratory analysis, walk through the proposed methodology, present analysis and results, and finally draw conclusions. An appendix with additional technical details and bibliography are also provided.

Data Sources and Exploratory Data Analysis

The Million Song Dataset (1) is a dataset with one million songs that was created as a collaboration between the Laboratory for the Recognition and Organization of Speech and Audio (LabROSA) and The Echo Nest. The Echo Nest has since been purchased by Spotify.

Initially, I downloaded the MSD dataset from the UCI Machine Learning Repository (2). From the one million songs in the dataset, it contained 515,345 observations (songs) since that was the number of

songs that contained the year (the response variable). This dataset had been processed so there was 1 response variable (year) and 90 predictor variables. The predictor variables were the 12 averages of the 12-dimensional timbre values and the 78 associated timbre covariances.

While this appeared to be an interesting dataset for predicting the year of a song and potentially for clustering, it wasn't going to allow for any natural language processing analysis since song title and artist were not included. As such, I needed to gather data on my own from the Million Song Database website (1).

Regardless, I attempted to use the full dataset (2) from UCI to try and predict the response, but I ran into issues when building models due to the large number of observations. Attempting to build models on my local machine was not possible for all models that I wanted to run as memory and computer power were inadequate. I would have needed to use cloud-based cluster computing for this size dataset.

Because of these computational complexity issues and because the UCI dataset didn't contain song names so was not going to be useful for NLP, I decided to use a subset of 10,000 song files from the Million Song Dataset website (1). This was approximately 2GB in size in a compressed format, as downloaded, and over 2.5GB after extraction. I had to read in the features from each file to create my own dataset which was smaller than the dataset from UCI and which contained song names.

The link for the subset of songs is at the MSD website (1) listed below, and it points to the compressed .gz file with all the song files.
http://millionsongdataset.com/pages/getting-dataset/#subset
http://labrosa.ee.columbia.edu/~dpwe/tmp/millionsongsubset.tar.gz (will start download)

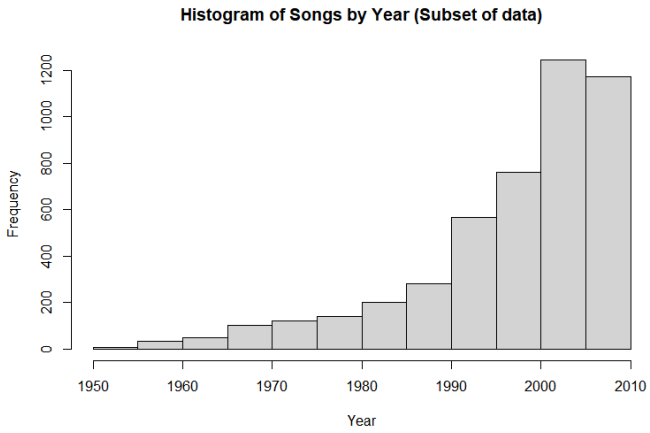
Using my dataset created from 10,000 songs, I inspected the dataset and found that 4680 songs contained a year, so I kept only these songs. The earliest year was 1926 and the most recent song was from 2010. There were 112 predictor variables and one response variable, year (see Appendix).

To inspect the distribution of songs, I assigned each song a "decade" classification based on the year the song was released and found the below distribution. The distribution is heavily skewed towards songs released more recently.

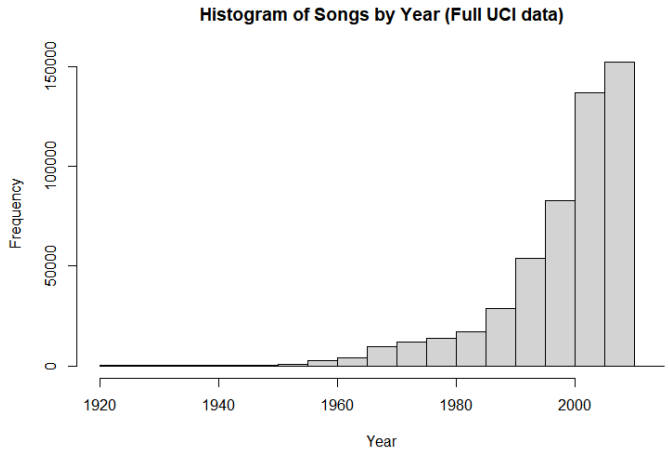
Decade	Count
1920	6
1930	6
1940	4
1950	30
1960	137
1970	246
1980	439
1990	1205
2000	2543
2010	64

Based on this distribution, I decided that since there were so few songs from pre-1950, I would not use those songs in my dataset. Although I could have used these songs to predict the year, I wanted to use the same dataset for all portions of my analysis. Since the number of songs (and thus, the number of terms in the song titles) was very small for the 1920s, 1930s, and 1940s, this meant that my natural language processing results would have been determined by only these few songs and wouldn't have been meaningful. Although the 2010 decade only contains songs from a single year, I decided to keep it because there were more songs (64 total songs from 2010).

After discarding the songs from pre-1950, I plotted a histogram of the song years to visualize the updated dataset.



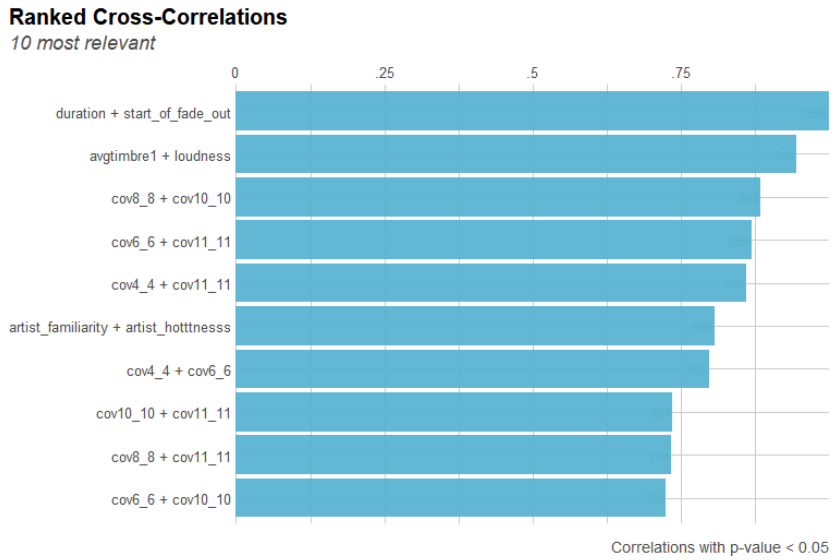
Since this was skewed towards songs from 1990 to 2010 (and especially songs from 2000 to 2010), I checked the UCI dataset (2) to see if it was more evenly spread and if the 10,000-song sample was exhibiting an unusual distribution relative to a larger dataset. As shown, the entire dataset from UCI (without song names or other additional features) had a similar distribution where there were very few songs from before 1950 and songs in the dataset were heavily unbalanced towards the 1990s and more recent. Because there were no large differences in distribution, I decided to continue with my smaller dataset without modifications.

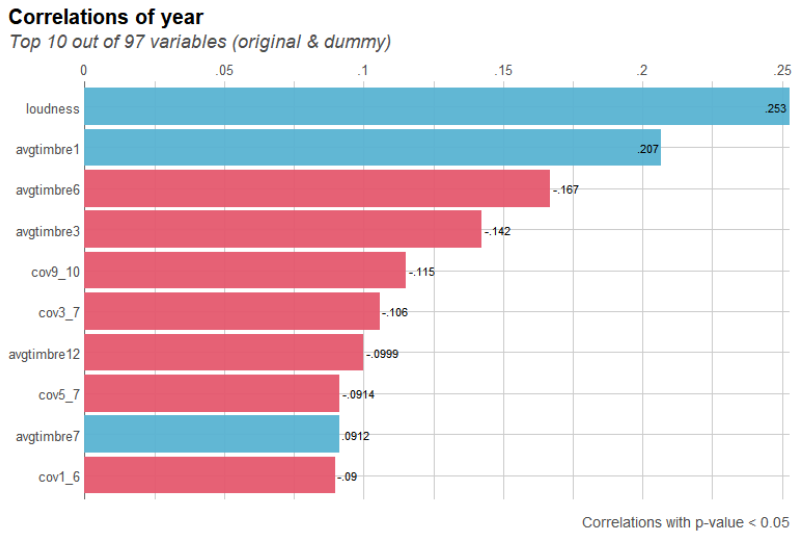


Based on these results I also decided that it made the most sense to treat the first part of my analysis as a regression problem rather than a classification problem. Initially, I planned to do a regression analysis to predict song year and I would also attempt a classification problem by year or decade. However, when doing a classification problem, imbalanced data is problematic and can lead to poor predictions for the underrepresented class(es).

There are methods for dealing with imbalanced data, such as undersampling or oversampling, but ultimately, I decided to leave the data as is and forego classification in favor of regression. The reason is because I am most interested in predicting the year and think this is more meaningful than the decade. As an example, predicting a song that is actually from 1980 as being from 1979 would misclassify the song if using a decade classification even though a prediction error of one year would be good. Similarly, if the song was predicted to be from 1989, it would be classified as correct if using decades even though the error is nine years so is far worse than a one-year error if 1979 was predicted.

Next, I looked for correlations in the variables. The strongest correlations were among predictor variables as shown below. The top correlations with the response (year) were mostly average timbre values and covariance timbre values, but the most correlated predictor was loudness. Since this variable was not included in the UCI dataset, this seems to show that building my own dataset which includes additional features will be beneficial.





Proposed Methodology

My methodology begins with data gathering and pre-processing.

First, as described previously, to get a more complete dataset than available from UCI (including song names and other details), I had to download a 10,000-song subset from the Million Song Database website (1). I also downloaded and attempted to use the UCI dataset (2) for comparison purposes, both to determine if I could create models on my local machine with the dataset (I could not; see previous section) and to check the distribution of songs.

For each of the 10,000 song files, I read the song information and metadata which included numerous features (see Appendix).

Although there was a “genre” feature, the dataset was incomplete in that not all songs had a listed genre. Even if genre was present for all songs, this seems like the type of feature that is prone to misclassification if for no other reason than many songs can fit into more than one genre. Since genre was not defined for all songs, I did not use it in my analysis.

Some features were unique identifiers for a given song or artist, either from the Echo Nest platform or another music website, so these should be removed. I noticed that the timbre average and covariance values from the UCI dataset were not provided in that format in the song files I downloaded; instead, timbre values were provided by segments throughout the duration of the song, so I had to compute my own timbre average and covariance values.

After gathering and cleaning this data, I wrote the dataset to a CSV file for easier processing moving forward. Using this data, I further cleaned up the songs and features (see feature/predictor variable list in the Appendix) I intended to use as discussed in the previous section.

Even after preprocessing to minimize the number of songs and features, I still had 4664 songs (observations) and 98 features. It can be computationally expensive to build cross-validated models with this many features, but regardless, I decided to build the following models. Before I started building models, I also scaled the predictor variables.

Linear Regression
Stepwise Regression (AIC)
Ridge
LASSO
Partial Least Squares (PLS)
Random Forest
Boosting

Year Prediction

I decided to use a 75/25% train/test split, and I used cross-validation to determine the best model and arrive at the most robust conclusion possible. For each iteration (with different train/test subsets), I recorded the test errors for each model and to come up with a final test error rate I averaged across all iterations. When training each model, I used the same random training/testing split for all models for a given iteration.

Because I expected the process to take a long time to run, I experimented by running only five iterations to start. Even only performing five train/test iterations, it took almost 30 minutes to run my code and predict with the seven above models. As a result, I decided to only run 30 iterations which took approximately three hours.

Linear and Stepwise Regression are straightforward in that there are no tuning parameters to consider when building these models. For Ridge and LASSO, for each train/test split, the optimal lambda value was chosen to ensure the best model was used when making predictions.

For PLS and Boosting, I used 10-fold cross-validation within each iteration. Since I am using Boosting for regression and not classification, I used a Gaussian distribution with 5000 trees.

For Random Forest, I used 32 random predictor variables (using a p/3 rule of thumb, where p = 97) as candidates for each split and grew 500 trees.

All models were built 30 times with different train/test splits as detailed above and predictions were made for all songs in the test subset for each model. Each calculated test error is the mean squared error (MSE) for the test predictions when compared to the true year that the song was released. The overall test error rate for each model was calculated as the average of all 30 individual MSE test error rates. The model with the lowest MSE will be deemed the best model, with possible consideration for the amount of variance.

Using cross-validation provides a much more robust and accurate overall test error rate since there is no concern that any given random split provided an outlier result where the test error rate was unusually high or low.

Clustering

When using k-means to cluster the songs. I built models with up to 15 clusters, and I used the silhouette method (see Appendix for discussion) to determine the optimal number of clusters. Initially, I attempted to use the elbow method, but the optimal number of clusters wasn't clear in my plot which is why I used

the silhouette method. I visualized the resulting clusters, and the results are detailed in the following section. Using the results, I will check to see if any groupings by year are evident.

Natural Language Processing

To determine what keywords are most indicative of a given decade, I used Term Frequency-Inverse Document Frequency (TF-IDF). This method looks at how common a term (a single word from a song title in this case) is in the overall corpus (all the song titles) as well as how frequently it appears in a certain class (decade) and computes a score to show how indicative the word is of a particular class. To compute the scores, I first tokenize (separate all the individual words from the song title) and remove stop words. Stop words are generally common words like "the, and, in, on," that are not informative when looking at different classes.

Analysis and Results

All analysis was done using R as a programming language and RStudio software.

Year Prediction

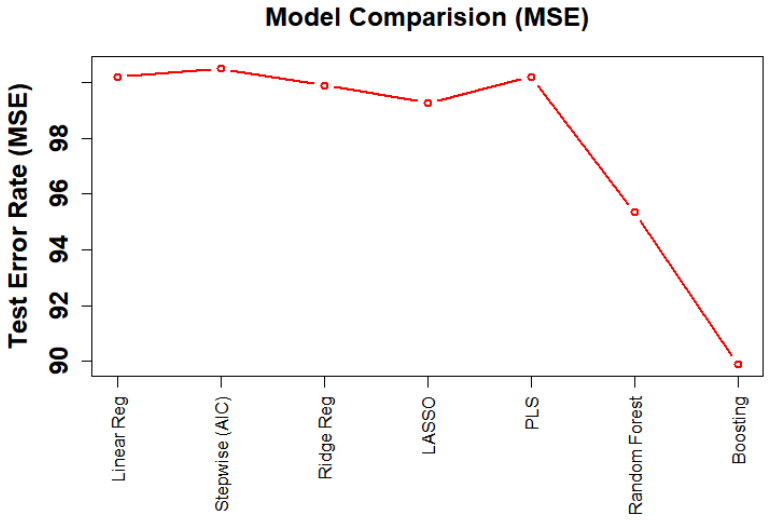
When predicting the year that a song was released, the seven different models provided the below mean squared errors (MSE). As shown, boosting provided the smallest test error and can be considered the best model. However, it had the worst variance.

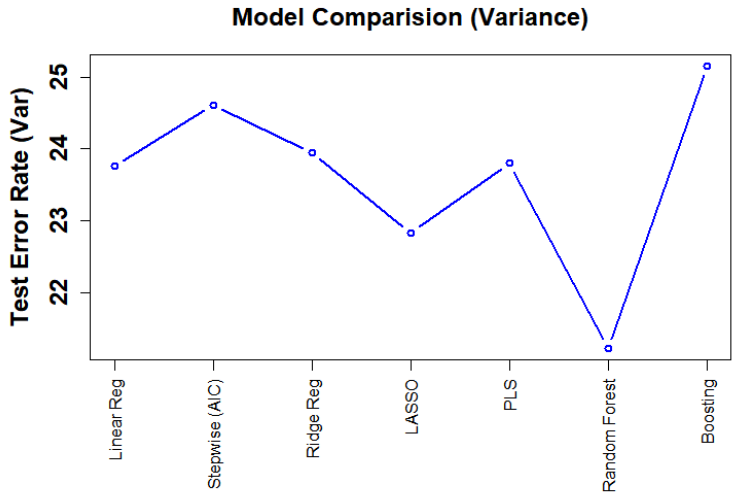
If the amount of variance in the boosting model was considered a problem, then the second-best model was Random Forest, and it also had the smallest variance so it would be the best choice.

Although boosting had the worst variance, it was similar to some of the other models and because it outperformed all other models for MSE by quite a lot, I still consider boosting to be the best model for predicting song years with this dataset.

The other five models had similar test errors to each other, but of those models LASSO was the best and Stepwise Regression with AIC was the worst.

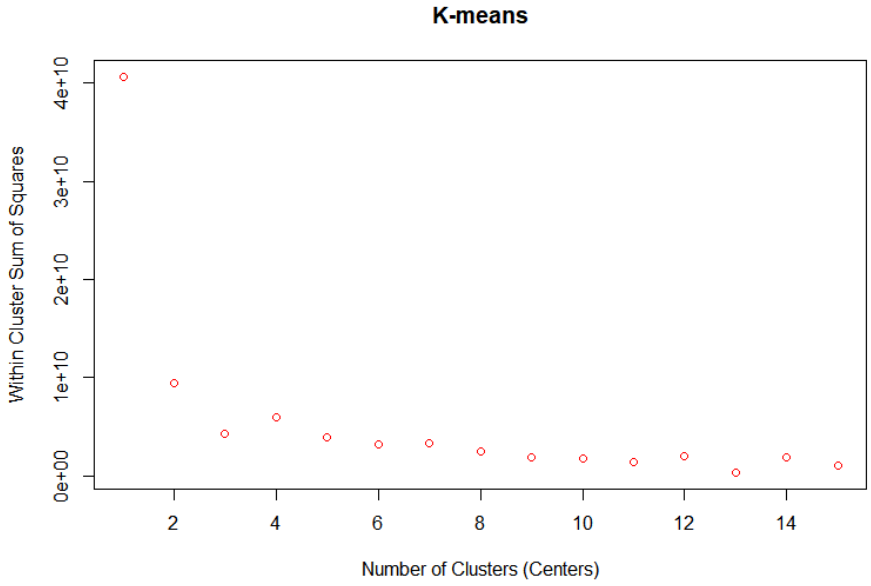
	Test Error (MSE)	Variance
Linear Regression	100.19	23.76
Stepwise Regression (AIC)	100.49	24.61
Ridge Regression	99.89	23.95
LASSO	99.26	22.84
Partial Least Squares (PLS)	100.19	23.80
Random Forest	95.34	21.23
Boosting	89.90	25.15

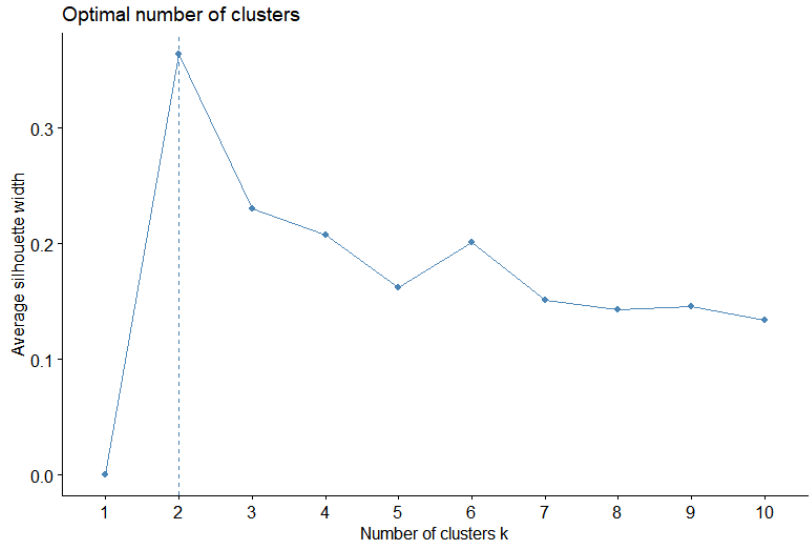




Clustering

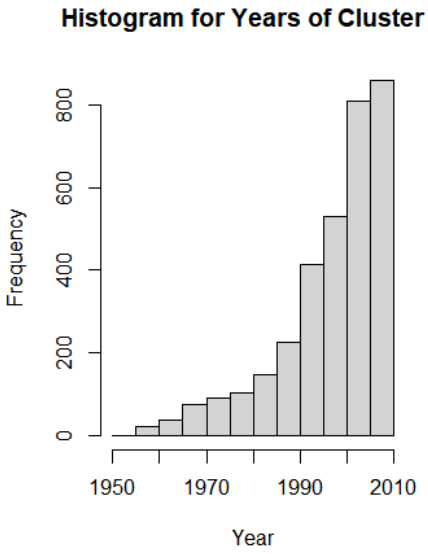
For K-means, I calculated the Within-Cluster-Sum-of-Squared-Errors and plotted them to find the optimal number of clusters. It was difficult to visually choose the optimal number of clusters from this plot using the elbow method because the values at k=3 and k=4 didn't create a smooth plot. As such, I used the silhouette method and found the optimal number of clusters was k=2 (see the Appendix for further discussion on choosing optimal clusters).

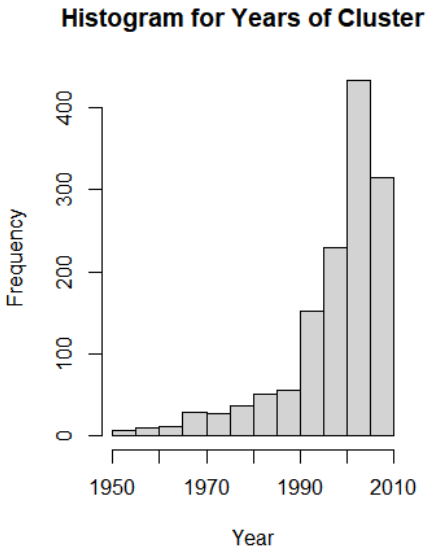


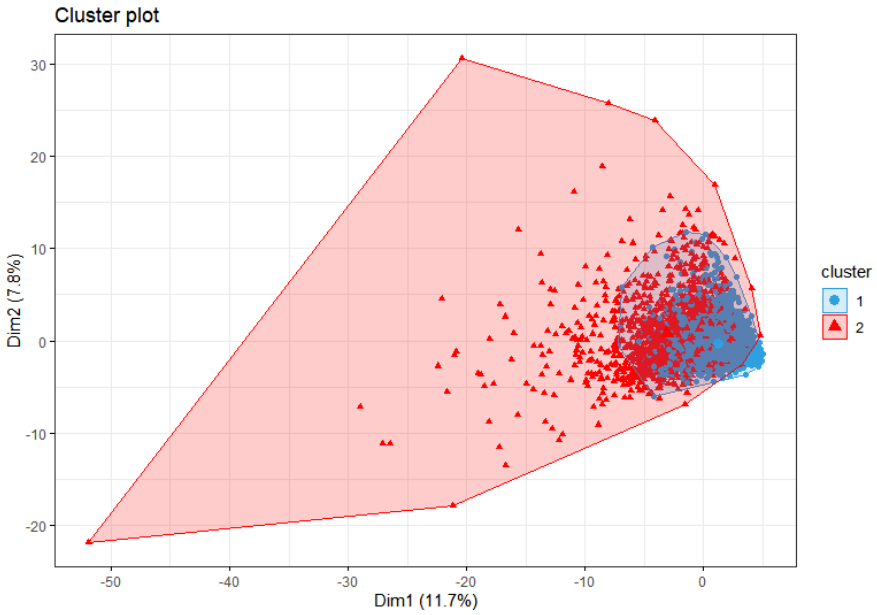


Using k=2 clusters, I wanted to understand if these two clusters separated the songs by some year (e.g., before and after 1990), so I created histograms and a cluster plot.

There are differences in the histograms in the number of observations by year, but there isn't a clear separation. In the cluster plot (which uses the top two principal components since there are more than two features), there is a lot of overlap which would further suggest that the songs are not distinct and well separated in the two clusters. As a result, clustering does not seem to be a good way to separate songs by year with the data that was gathered.







Natural Language Processing

Using TF-IDF, I found the top 10 words per decade. For the 1950s and 2010s, there were relatively few songs and as a result there were a lot of ties, which meant that some top words only appeared in the decade once. As a result, there were more than 10 songs for these decades. See the Appendix for details on how certain song title situations (e.g., duplicate words or songs) were handled.

A selection of interesting “Top 10” words is provided below (full list in the Appendix).

1950s – carnaval, cantando, conguitos, faith

1960s – watermelon, quiet, spoken, word, lp

1970s – bit, bully, version, digital, remaster, lp, waste

1980s – start, version, make, night, Halloween, windpower

1990s – version, lp, live, explicit, red, album, mix

2000s – remix, version, live, amended, album, explicit, featuring, edit, mix, club

2010s – feat, diamond, cannonball, minion, twins, underworld

As shown, there were several words that made the Top 10 list even though they were important in more than one decade. Some examples are:

version, lp, album, explicit

I decided not to add these to my stop words list because I thought it was interesting to see how these terms change over time. For example, we start seeing the words “explicit” and “live” in song titles in the 1990s, which seems reasonable if you listened to alternative rock in the late 1990s. Songs also started “mix”-ing in the 1990s and “remix”-ing in the 2000s.

Similarly, we start seeing the words “featuring” and “feat” in the 2000s and 2010s. Although artists have collaborated for a long time, it seems like song titles regularly started saying that the song featured another artist in the 2000s. And on the other hand, the term “lp” is no longer important after the 1990s as might be anticipated.

Notably, several of the top words from the 1950s are in other languages, whereas this is less common in other decades. And the 1970s made use of terms like “bit,” “digital,” and “remaster,” as music moved from analog to digital.

Many of these observations from different decades for high-value words are understandable if you lived through or understand music in these decades.

For fun, some surprising terms are “watermelon” which was tied for the more important word in the 1960s. “Bully” and “waste” were top 5 words in the 1970s, while “Halloween” and “windpower” were top 10 words in the 1980s and “club” was a top word in the 2000s (not overly surprising). The 2010s had a lot of ties, so although many top words only appeared once, it’s entertaining to see “minion,” “twins,” and “underworld” tied for top words even if these may not be very meaningful due to the small number of songs.

Conclusion

In conclusion, boosting provided the best model with the lowest test error rate. The second-best model was random forest which also had the smallest variance, as opposed to boosting which had the worst variance.

However, the best prediction score was an MSE of 89.9, which means the average error was approximately 9.5 years. One possible reason the test error wasn’t better may be because music often

changes incrementally. As such, it is difficult to predict what year a song is from with a high level of accuracy.

Using K-means clustering didn't provide a meaningful separation of songs, at least not that was correlated with year. If other classifications were available, it might have been interesting to see if clusters corresponded to something else.

Natural Language Processing provided some interesting high-value words, in some cases allowing us to draw conclusions about music trends over time.

Possible future work includes gathering more complete information for some discarded features to see if they provide additional predictive power. For example, if Latitude and Longitude details were available for all artists and songs, then if music was coming out of a small region or similar area during a particular time period, this might improve the predictive power of the models.

More complete information could also include gathering genre classifications for all songs and seeing if these lead to improved year predictions. K-means clustering models could also be updated to see if clusters indicate time periods once genre is included. Alternately, models could be built that included year and then compared to genre labels to see if clustering is indicative of genre.

One final future project could be to predict the years for all songs in an album and find the mean or median of these songs to see if this leads to improved prediction.

Appendix

As-Downloaded Features/Predictor Variables

Note: Certain variables like “segments timbre” are arrays and contain lots of information. From this array in particular, I extracted 90 features.

Field name	Type	Description
analysis sample rate	float	sample rate of the audio used
artist 7digitalid	int	ID from 7digital.com or -1
artist familiarity	float	algorithmic estimation
artist hotttnesss	float	algorithmic estimation
artist id	string	Echo Nest ID
artist latitude	float	latitude
artist location	string	location name
artist longitude	float	longitude
artist mbid	string	ID from musicbrainz.org
artist mbtags	array string	tags from musicbrainz.org
artist mbtags count	array int	tag counts for musicbrainz tags
artist name	string	artist name
artist playmeid	int	ID from playme.com, or -1
artist terms	array string	Echo Nest tags
artist terms freq	array float	Echo Nest tags freqs
artist terms weight	array float	Echo Nest tags weight
audio md5	string	audio hash code
bars confidence	array float	confidence measure
bars start	array float	beginning of bars, usually on a beat
beats confidence	array float	confidence measure
beats start	array float	result of beat tracking
danceability	float	algorithmic estimation
duration	float	in seconds
end of fade in	float	seconds at the beginning of the song
energy	float	energy from listener point of view
key	int	key the song is in
key confidence	float	confidence measure
loudness	float	overall loudness in dB
mode	int	major or minor
mode confidence	float	confidence measure
release	string	album name
release 7digitalid	int	ID from 7digital.com or -1
sections confidence	array float	confidence measure

sections start	array float	largest grouping in a song, e.g. verse
segments confidence	array float	confidence measure
segments loudness max	array float	max dB value
segments loudness max time	array float	time of max dB value, i.e. end of attack
segments loudness max start	array float	dB value at onset
segments pitches	2D array float	chroma feature, one value per note
segments start	array float	musical events, ~ note onsets
segments timbre	2D array float	texture features (MFCC+PCA-like)
similar artists	array string	Echo Nest artist IDs (sim. algo. unpublished)
song hotttnesss	float	algorithmic estimation
song id	string	Echo Nest song ID
start of fade out	float	time in sec
tatums confidence	array float	confidence measure
tatums start	array float	smallest rythmic element
tempo	float	estimated tempo in BPM
time signature	int	estimate of number of beats per bar, e.g. 4
time signature confidence	float	confidence measure
title	string	song title
track id	string	Echo Nest track ID
track 7digitalid	int	ID from 7digital.com or -1
year	int	song release year from MusicBrainz or 0

Features/Predictor Variables List

artist_familiarity	artist_hotttnesss	year	avgtimbre1	avgtimbre2	avgtimbre3
avgtimbre4	avgtimbre5	avgtimbre6	avgtimbre7	avgtimbre8	avgtimbre9
avgtimbre10	avgtimbre11	avgtimbre12	cov1_1	cov1_2	cov1_3
cov1_4	cov1_5	cov1_6	cov1_7	cov1_8	cov1_9
cov1_10	cov1_11	cov1_12	cov2_2	cov2_3	cov2_4
cov2_5	cov2_6	cov2_7	cov2_8	cov2_9	cov2_10
cov2_11	cov2_12	cov3_3	cov3_4	cov3_5	cov3_6
cov3_7	cov3_8	cov3_9	cov3_10	cov3_11	cov3_12
cov4_4	cov4_5	cov4_6	cov4_7	cov4_8	cov4_9
cov4_10	cov4_11	cov4_12	cov5_5	cov5_6	cov5_7
cov5_8	cov5_9	cov5_10	cov5_11	cov5_12	cov6_6
cov6_7	cov6_8	cov6_9	cov6_10	cov6_11	cov6_12
cov7_7	cov7_8	cov7_9	cov7_10	cov7_11	cov7_12
cov8_8	cov8_9	cov8_10	cov8_11	cov8_12	cov9_9
cov9_10	cov9_11	cov9_12	cov10_10	cov10_11	cov10_12
cov11_11	cov11_12	cov12_12	duration	end_of_fade_in	loudness
start_of_fade_out	tempo				

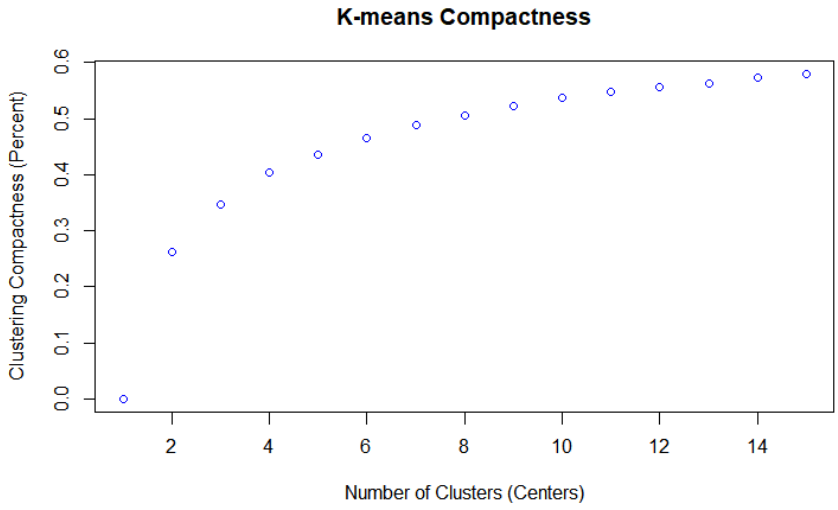
Cross-Validation Output (By Iteration and Model)

	Linear Reg	Stepwise (AIC)	Ridge Reg	LASSO	PLS	Random Forest	Boosting
1	99.61827	100.96554	99.24259	98.62065	99.63181	97.54209	91.69286
2	94.62909	95.81801	93.22512	91.70836	94.62916	87.56224	82.86494
3	100.42717	99.76078	100.11056	99.51877	100.42635	96.42600	92.53932
4	101.63829	101.71733	101.38344	100.91571	101.68073	96.65452	95.49185
5	99.37125	99.54979	98.80459	97.91183	99.35474	91.77144	87.44351
6	99.85952	100.50841	99.52366	98.97922	99.86067	96.23131	92.47989
7	101.58432	102.67768	101.54640	101.10244	101.50462	95.74888	87.15128
8	92.13918	92.91174	91.89394	91.58282	92.10216	90.05406	81.83394
9	103.50832	103.12682	102.64786	101.69064	103.53268	98.22379	90.06187
10	98.54752	98.67682	98.41756	97.97730	98.54670	94.50756	88.62226
11	105.60651	104.73282	105.19562	103.24053	105.63113	96.40797	91.76959
12	100.14625	100.20971	100.80689	100.01179	100.14101	95.05536	88.44147
13	96.02639	95.10443	94.98609	94.53789	96.04967	90.35168	83.28995
14	98.07142	98.39330	97.67552	97.39637	98.07923	95.36885	91.24735
15	99.07003	98.90992	98.97073	98.28886	99.05553	98.15266	92.22720
16	103.05878	102.34047	102.95290	101.75813	103.06777	92.72108	88.39985
17	93.69323	93.32081	93.95942	92.60612	93.69466	88.98889	82.42245
18	106.50571	106.75351	106.13268	105.39518	106.45421	99.86779	94.82792
19	107.20810	108.48354	106.85338	106.59199	107.18879	105.77397	97.71098
20	96.13448	95.72952	95.72750	95.40008	96.13489	91.27808	87.35217
21	110.79029	112.37901	110.97459	111.10547	110.79033	106.39019	101.82617
22	105.84171	105.77670	105.81822	105.39397	105.84170	103.44067	97.81125
23	99.08632	101.54872	98.59115	98.01700	99.09955	94.24361	91.54639
24	105.26035	105.27293	104.81509	104.34824	105.29340	95.53214	93.81180
25	101.20601	100.86046	100.42289	99.58628	101.22630	94.90238	89.77351
26	105.50815	106.40483	105.16783	103.43836	105.59626	98.28832	92.41568
27	89.01767	90.14848	89.33352	90.57156	89.01753	88.00471	80.67736
28	95.22153	95.49167	95.42036	95.39418	95.21667	92.04400	87.12559
29	100.66739	101.19261	100.42899	99.54021	100.68019	95.89827	89.01068
30	96.24708	95.92820	95.67808	95.15204	96.24708	92.71941	85.19569

K-means Discussion

To determine the optimal number of clusters to group the data into, I intended to use the within-cluster-sum-of-squares and look for the elbow point. However, when I looked at the plot, there wasn't a clear transition point. As a result, I plotted a similar measure, the compactness, but it didn't provide a clear transition point either.

I decided to use the silhouette method which provides a value to show how good the clustering is. As shown in the results section, k=2 was the optimal number of clusters when using the silhouette method.



Natural Language Processing Discussion

I ran into a couple of unexpected situations when analyzing the data. In at least one instance a song title was given in a foreign language with a translation in parentheses, all in the song title field. I considered removing the duplication, but this appears to have been the title as released on an English language album, so I left it alone. One effect this has is that when an uncommon word is duplicated in both languages, it gets counted twice for term frequency. Regardless, my intention was not to modify song titles from how they were released so I chose to let any double counts remain.

I also found some songs were released on one album and then again on a greatest hits album. Similarly, I found some songs that were released and then another instrumental version was released with very similar names (e.g., Again; Again (Instrumental); by Janet Jackson). Since these are technically different releases, and even different songs, I left them in my analysis even though these types of events serve to increase certain term frequencies.

TF-IDF Top 10 Terms (by Decade)

decade	tokens	tf_idf
1950	carnaval	0.052
1950	arbre	0.027
1950	auprã	0.027
1950	cantando	0.027
1950	colombine	0.027
1950	conguitos	0.027
1950	darling	0.027
1950	faith	0.027
1950	film	0.027
1950	freight	0.027
1950	hier	0.027
1950	hound	0.027
1950	law	0.027
1950	manuel's	0.027
1950	mãºsicas	0.027
1950	molly	0.027
1950	moulin	0.027
1950	orfeo	0.027
1950	rouge	0.027
1950	sophisticated	0.027
1950	spell	0.027
1950	unlocks	0.027
1950	wanderin	0.027
1960	autres	0.012
1960	corcovado	0.012
1960	watermelon	0.012
1960	est	0.008
1960	jack	0.008
1960	quiet	0.008
1960	spoken	0.008
1960	word	0.008
1960	will	0.007
1960	lp	0.006
1970	bit	0.013
1970	lp	0.009
1970	bully	0.007
1970	concerto	0.007
1970	waste	0.007
1970	version	0.006
1970	digital	0.006
1970	remaster	0.006
1970	little	0.005
1970	piece	0.004
1980	start	0.008
1980	version	0.004
1980	make	0.004
1980	night	0.004
1980	canta	0.004
1980	chim	0.004
1980	halloween	0.004
1980	raise	0.004
1980	says	0.004
1980	windpower	0.004

1990	version	0.005
1990	lp	0.004
1990	live	0.004
1990	explicit	0.004
1990	get	0.003
1990	red	0.003
1990	album	0.003
1990	mix	0.002
1990	know	0.002
1990	un	0.002
2000	remix	0.006
2000	version	0.005
2000	live	0.004
2000	amended	0.004
2000	album	0.004
2000	mix	0.003
2000	explicit	0.003
2000	featuring	0.003
2000	edit	0.003
2000	club	0.003
2010	feat	0.016
2010	diamond	0.015
2010	ad	0.012
2010	assis	0.012
2010	baixas	0.012
2010	begins	0.012
2010	bilal	0.012
2010	brockington	0.012
2010	bruno	0.012
2010	burp	0.012
2010	cannonballs	0.012

2010	catharsis	0.012
2010	chances	0.012
2010	clamamus	0.012
2010	colpa	0.012
2010	creatures	0.012
2010	cured	0.012
2010	daddy's	0.012
2010	darien	0.012
2010	descontrol	0.012
2010	desilusiã	0.012
2010	disappearing	0.012
2010	documents	0.012
2010	doug	0.012
2010	emergency	0.012
2010	ett	0.012
2010	evenfall	0.012
2010	evigne	0.012
2010	exsvles	0.012
2010	far	0.012
2010	gimme	0.012
2010	haggard	0.012
2010	heretic	0.012
2010	ilbert	0.012
2010	jail	0.012
2010	jest	0.012
2010	kui	0.012
2010	legs	0.012
2010	liberi	0.012
2010	mars	0.012
2010	mazurcas	0.012
2010	mccomb	0.012

2010	milladoiro	0.012
2010	minion	0.012
2010	mortva	0.012
2010	nature's	0.012
2010	noch	0.012
2010	norrskensdã	0.012
2010	oceans	0.012
2010	para	0.012
2010	parasitic	0.012
2010	per	0.012
2010	quã	0.012
2010	rãas	0.012
2010	rebeneb	0.012
2010	rette	0.012
2010	sabotage	0.012
2010	shell	0.012
2010	spã	0.012
2010	taevas	0.012
2010	ted	0.012
2010	telepathy	0.012
2010	ter	0.012
2010	threw	0.012
2010	trick	0.012
2010	tua	0.012
2010	twins	0.012
2010	underworld	0.012
2010	vitalized	0.012
2010	wigue	0.012
2010	wir	0.012
2010	wollen	0.012
2010	won	0.012

Bibliography and Credits

1. Million Song Dataset, official website by Thierry Bertin-Mahieux, available at: http://millionsongdataset.com/

2. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository. “Year Prediction MSD“ [https://archive.ics.uci.edu/ml/datasets/yearpredictionmsd, http://archive.ics.uci.edu/ml], Irvine, CA: University of California, School of Information and Computer Science.

3. Bertin-Mahieux, Thierry, Daniel PW Ellis, Brian Whitman, and Paul Lamere. "The million song dataset." (2011): 591-596.

4. Jayaprakash Nallathambi. “R Series — K Means Clustering (Silhouette) - CodeSmart - Medium.” Medium. CodeSmart, June 18, 2018. https://medium.com/codesmart/r-series-k-means-clustering-silhouette-794774b46586.

5. Fonseca, Luiz. “Clustering Analysis in R Using K-Means - towards Data Science.” Medium. Towards Data Science, August 15, 2019. https://towardsdatascience.com/clustering-analysis-in-r-using-k-means-73eca4fb7967.