Predicting song trends in Denver, Colorado

Zain Elsell, Avery Fulton, Ayush Uniyal

GitHub: https://github.com/Vizim/spotify_data_proj

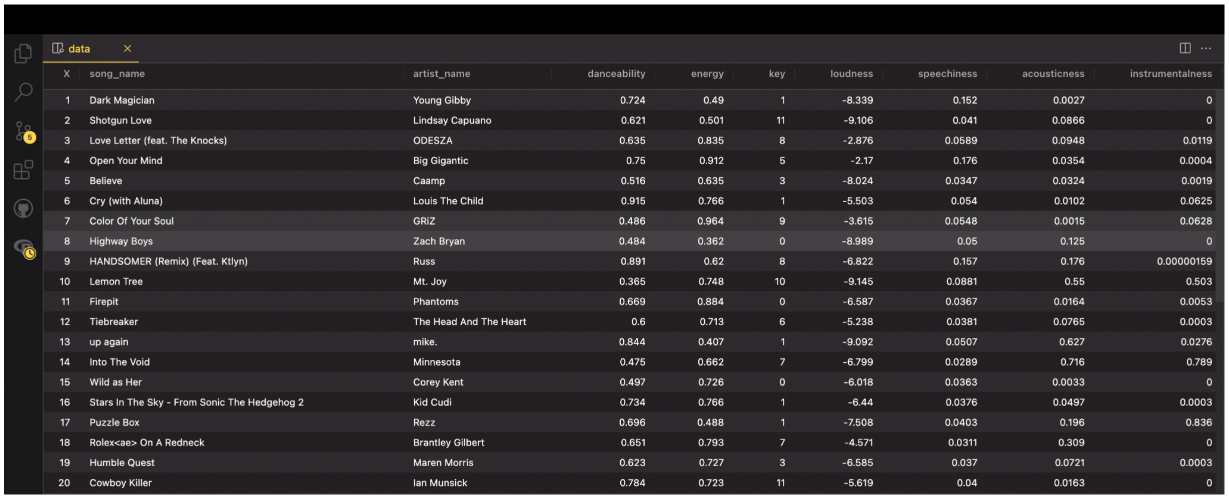
It's undeniable that music plays an enormous role in our everyday lives. Now, this isn't just hearsay. According to a study conducted by Marie Charlotte Götting on music listening habits in the United States, 68% of adults aged 18-38 stated that they listen to music every single day. This isn't surprising considering that the music industry generates billions of dollars in revenue every year. In the past, if we asked what makes a particular song popular, the type of answers given would have been speculative at best, but luckily, how music is consumed and distributed has drastically shifted in the last ten years due to the rise of streaming platforms.

Streaming platforms that provide quantitative data allow them to rank songs based on the total number of plays. This brings us to our project; our primary goal was to predict the trending ability of any particular song. More specifically, we chose to restrict the scope of our analysis to songs that are uniquely popular in Denver, Colorado, to showcase the local music taste rather than the national one. The primary source of our data was the Spotify Charts website, particularly the Denver Local Pulse Chart (DLPC). This chart shows the top 100 songs uniquely popular in Denver every week. After scraping the DLPC for 24 weeks and in conjunction with the Spotify API, we extracted a variety of information on each track. Since the data was collected outside of the context of a controlled experiment, all of our data is observational. Our goals throughout the analysis were to identify what quantitative values affect a song's popularity and can we use those values to create machine learning models that are capable of predicting a song's popularity based on track information.

Data Acquisition

Using the Spotify API, we identify specific audio features indicative of a track with a high chart ranking. Some of the features we examined include: features of song name, artist name, danceability, energy, key, speechiness, acousticness, instrumentalness, liveness, valence, tempo, track identifier, duration, time signature, week scraped, and song ranking. We generated a sequence of dates beginning in late October and ending at the end of March to start the data acquisition process. This range of dates was the maximum that we could view on the DLPC. The dates in our sequence were then converted from an R list to a Python list and then piped into a set of Python scripts that would be able to read to help simplify our data acquisition process. In short, the scripts copied the HTML table class on the DLPC to a text file and then switched the week to the next in our sequence.

While this may seem like a very roundabout solution, it was the only viable option since the DLPC requires a Spotify account to view and additionally has lots of anti-scraping protocols in place, such as captchas. After we had our text file, we started to interact with the Spotify web API. To do this, we used the SpotifyR wrapper class. Using a function that would pull the song IDs directly from the raw HTML and put them into a list, we could then interact with the Spotify API, getting us all the vital information about that specific song, finally generating our data frame.



data

X	song_name	artist_name	danceability	energy	key	loudness	speechiness	acousticness	instrumentalness
1	Dark Magician	Young Gibby	0.724	0.49	1	-8.339	0.152	0.0027	0
2	Shotgun Love	Lindsay Capuano	0.621	0.501	11	-9.106	0.041	0.0866	0
3	Love Letter (feat. The Knocks)	ODESZA	0.635	0.835	8	-2.876	0.0589	0.0948	0.0119
4	Open Your Mind	Big Gigantic	0.75	0.912	5	-2.17	0.176	0.0354	0.0004
5	Believe	Caamp	0.516	0.635	3	-8.024	0.0347	0.0324	0.0019
6	Cry (with Aluna)	Louis The Child	0.915	0.766	1	-5.503	0.054	0.0102	0.0625
7	Color Of Your Soul	GRiZ	0.486	0.964	9	-3.615	0.0548	0.0015	0.0628
8	Highway Boys	Zach Bryan	0.484	0.362	0	-8.989	0.05	0.125	0
9	HANDSOMER (Remix) (Feat. Ktlyn)	Russ	0.891	0.62	8	-6.822	0.157	0.176	0.00000159
10	Lemon Tree	Mt. Joy	0.365	0.748	10	-9.145	0.0881	0.55	0.503
11	Firepit	Phantoms	0.669	0.884	0	-6.587	0.0367	0.0164	0.0053
12	Tiebreaker	The Head And The Heart	0.6	0.713	6	-5.238	0.0381	0.0765	0.0003
13	up again	mike.	0.844	0.407	1	-9.092	0.0507	0.627	0.0276
14	Into The Void	Minnesota	0.475	0.662	7	-6.799	0.0289	0.716	0.789
15	Wild as Her	Corey Kent	0.497	0.726	0	-6.018	0.0363	0.0033	0
16	Stars In The Sky - From Sonic The Hedgehog 2	Kid Cudi	0.734	0.766	1	-6.44	0.0376	0.0497	0.0003
17	Puzzle Box	Rezz	0.696	0.488	1	-7.508	0.0403	0.196	0.836
18	Rolex<ae> On A Redneck	Brantley Gilbert	0.651	0.793	7	-4.571	0.0311	0.309	0
19	Humble Quest	Maren Morris	0.623	0.727	3	-6.585	0.037	0.0721	0.0003
20	Cowboy Killer	Ian Munsick	0.784	0.723	11	-5.619	0.04	0.0163	0

Fig 1. Part of our final rendered data frame can be found on GitHub under clean_data/track_data.csv

Data Analysis

Features and Correlations:

To help identify which specific audio features were being used with a "high ranked" song, we created a correlation matrix of all the numeric features in our data set—looking for strong correlations between rank and audio features. When assessing the correlations, they could've been either positive or negative. For our purposes, we generally want to find negative correlations since the lower a song's rank is, the closer it is to that number 1 spot. Looking at the far-right column of our dataset, we can see a positive correlation between rank and danceability, along with rank and valence. After further analyzing the overall correlation between rank and audio features, we concluded that energy, loudness, instrumentalness, liveliness, and duration were the audio features that were most correlated with a song being ranked highly. To help balance our model, we added audio features that negatively correlated with our model to help us organize poorly ranking songs.

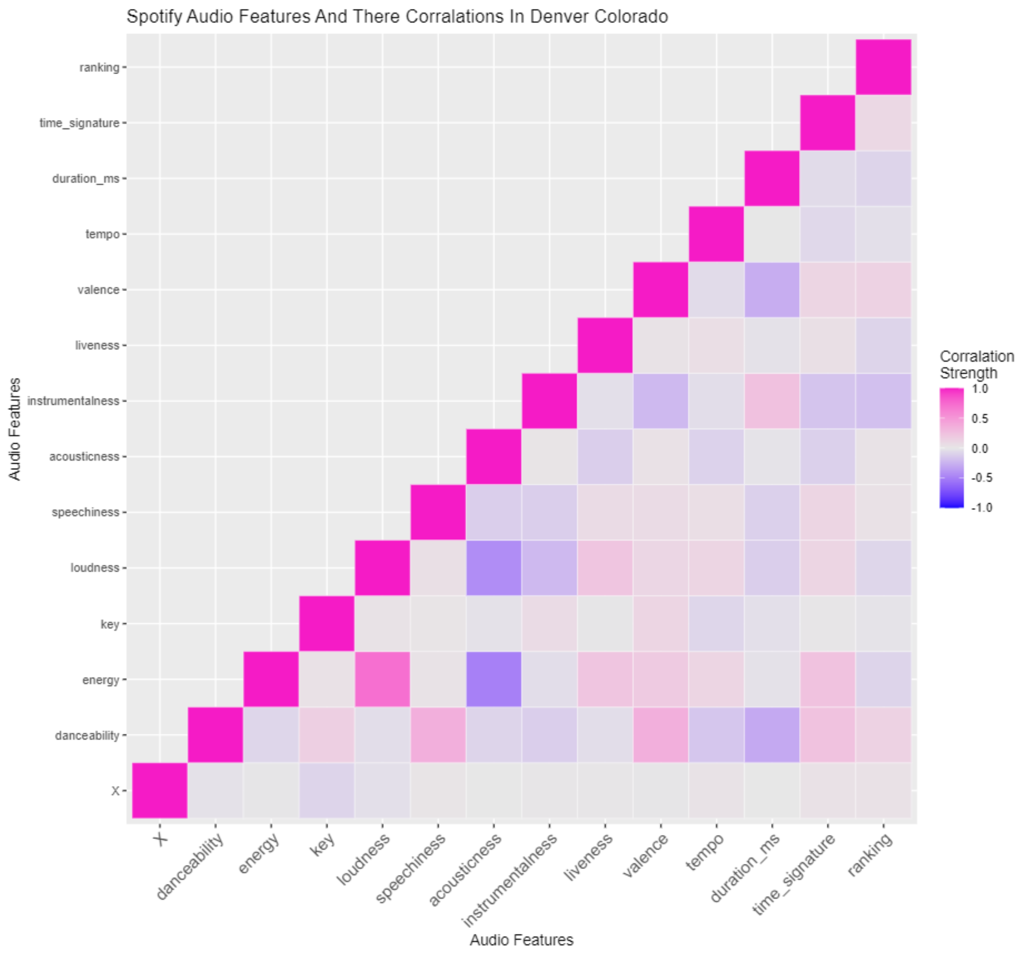


Fig 2. Correlation matrix for the relevance of track audio features. High correlation coefficients are shown in pink, while low correlation coefficients are shown in purple.

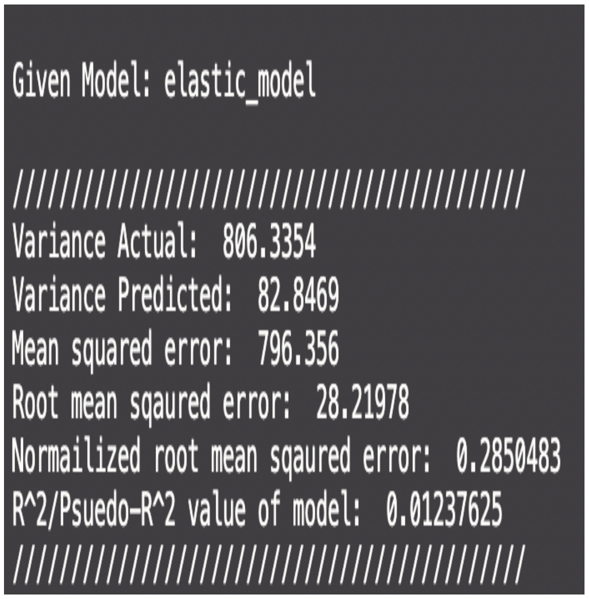
Model Testing

We primarily relied on a couple of different metrics to test our models. First, we wrote a function to represent our rankings as binary numbers with a cut-off point. We then put in our actual ranking vs. our predicted rankings from each model in their binary representations. We generated a confusion matrix on the set using the test set exclusively. Next, we needed to define our testing metrics. Some of these include: the variance of our actual data (ranking), the variance of our predicted data, our mean squared error (MSE), the root mean squared error (RMSE), the normalized root mean squared error (NRME), and finally, the R-Squared Value.(When using the random forest model, we used the following $\sqrt{\frac{MSE}{Variance\ Acutal\ Set}}$ to evaluate accuracy)

Lastly, we partitioned our data into a training and test set with a 70:30 split after identifying our correlations. When calculating our binary test of the top X amount of tracks, we need to make sure that the distribution of our test isn't skewed. Using a histogram of our test set and randomly sampling, we found that 30% of the test set data was slightly skewed towards worse ranking tracks. In the future, it would be best to sample a normal distribution of values for our test set to represent the data in our binary conversions better.

LASSO Regression:

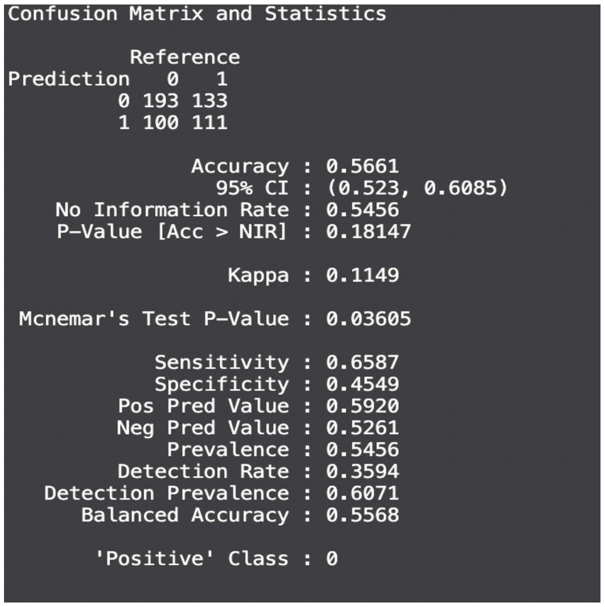
Now that we had a complete data set and test cases, we could move on to our next step of creating models. We needed to figure out what machine learning model would help us best predict a song's popularity in Denver. Our first approach was using a LASSO regression model. The reason why we initially chose this model was to automate feature selection. We wanted to automate feature selection because, at the time, we weren't sure what features had the strongest predictive pull on our model. LASSO regression works the best when there are lots of correlated features. The problem we found was that the features didn't have as strong a pull as we initially hypothesized. This led to all of the coefficients within our model being pushed to zero. In turn, this produced an extremely linear spread of our data. This is reflected in our meager R-squared score of 0.07. Additionally, we attempted to alter our alpha value of the model to use an Elastic Net Model and a Ridge Regression model, but this led to equally poor results with an R-squared score of 0.07 and a binary score of 56 with a 50% cut-off.



```
Given Model: elastic_model

/////////////////////////////////////////////
Variance Actual:  806.3354
Variance Predicted:  82.8469
Mean squared error:  796.356
Root mean sqaured error:  28.21978
Normailized root mean sqaured error:  0.2850483
R^2/Psuedo-R^2 value of model:  0.01237625
/////////////////////////////////////////////
```

Fig 3. The metrics output from our elastic net model.



```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 193 133
         1 100 111

               Accuracy : 0.5661
                 95% CI : (0.523, 0.6085)
    No Information Rate : 0.5456
    P-Value [Acc > NIR] : 0.18147

                  Kappa : 0.1149

 Mcnemar's Test P-Value : 0.03605

            Sensitivity : 0.6587
            Specificity : 0.4549
         Pos Pred Value : 0.5920
         Neg Pred Value : 0.5261
             Prevalence : 0.5456
         Detection Rate : 0.3594
   Detection Prevalence : 0.6071
      Balanced Accuracy : 0.5568

       'Positive' Class : 0
```

Fig 4. The confusion matrix output from our elastic net regression model.

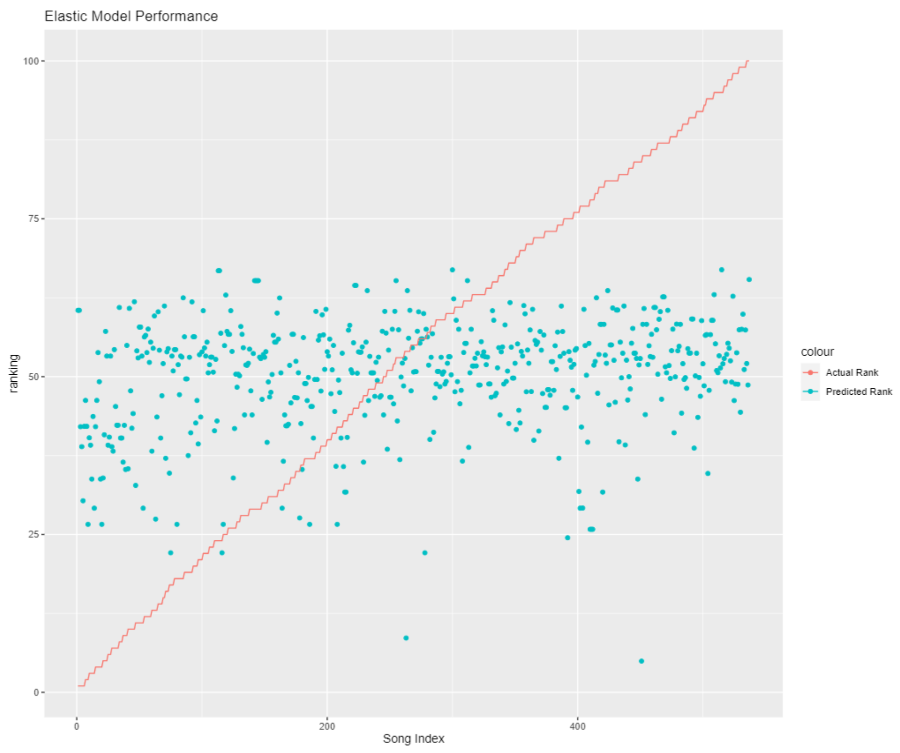


Fig 5.
As you can see, the predicted values are linearly spread between ranks 25 through 70, meaning that our values did not correspond to the actual ranking.

Random Forest Regression:

The second regression model that we decided on was the random forest regression. This approach worked significantly better than our initial elastic net regression, as shown in Figures 6,7,8 and 9. This particular model uses 500 regression trees and, in total, generates each tree ten times, picking the lowest RMSE of the ten attempts as the final model. Overall, Fig 7 shows that our testing data set fits better with our predicted data set significantly better. Additionally, as seen in Fig 8, we were able to explain about 80% of the overall variance of the test set using our "Psuedo R-squared Metric ."The Binary test metric, as seen in Fig 9, also improved, showing that 70% of the time, the random forest model could predict that a data point was located in the top 50 tracks.

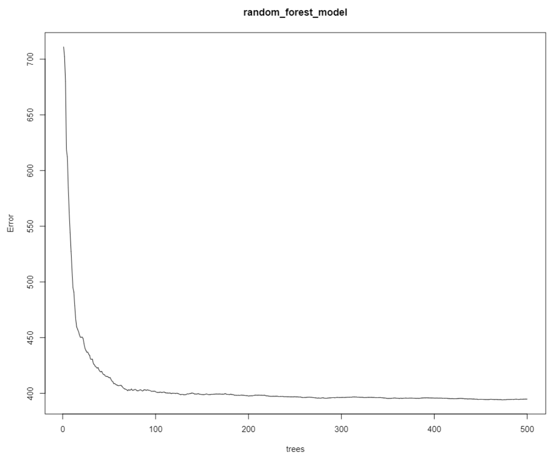


Fig 6. This shows the effect of random forest modeling where the RMSE shrinks as the number of trees increases

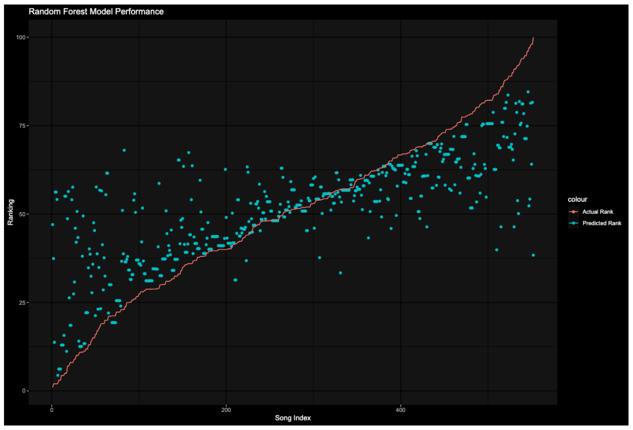
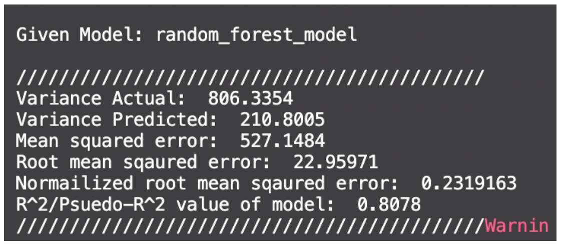


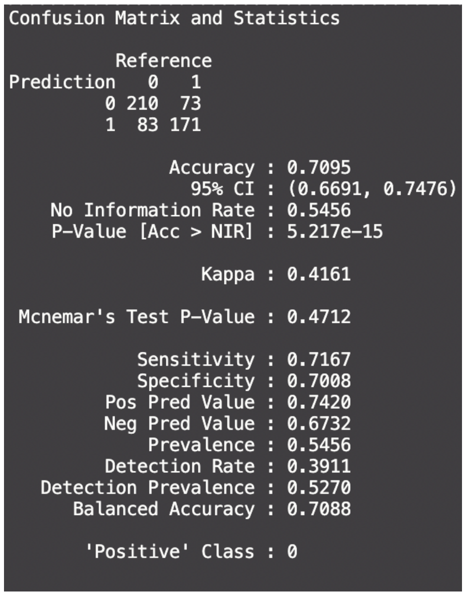
Fig 7. The predicted values better aligned with the actual ranking compared to the



```
Given Model: random_forest_model

///////////////////////////////////////////
Variance Actual:  806.3354
Variance Predicted:  210.8005
Mean squared error:  527.1484
Root mean sqaured error:  22.95971
Normailized root mean sqaured error:  0.2319163
R^2/Psuedo-R^2 value of model:  0.8078
///////////////////////////////////////////Warnin
```

Fig 8. The metrics output from our elastic net model.



```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 210  73
         1  83 171

               Accuracy : 0.7095
                 95% CI : (0.6691, 0.7476)
    No Information Rate : 0.5456
    P-Value [Acc > NIR] : 5.217e-15

                  Kappa : 0.4161

 Mcnemar's Test P-Value : 0.4712

            Sensitivity : 0.7167
            Specificity : 0.7008
         Pos Pred Value : 0.7420
         Neg Pred Value : 0.6732
             Prevalence : 0.5456
         Detection Rate : 0.3911
   Detection Prevalence : 0.5270
      Balanced Accuracy : 0.7088

       'Positive' Class : 0
```

Fig 9. The confusion matrix output from our random forest model.

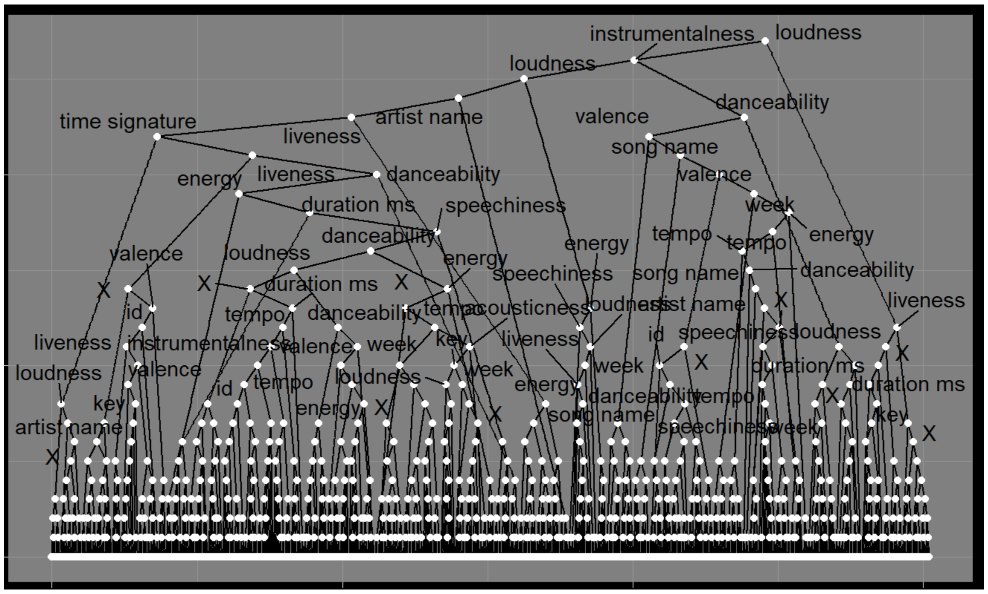


Fig 10. Snapshot 1 of the 10 regression of the tree generated by the Random Forrest model.

Conclusion & Future Concicderations

During our study, we concluded that audio features alone aren't very viable for rank. If given a chance to continue this analysis, reducing the noise in our data set would've likely improved our model accuracy. We had a lot of positive and negative data points in our data set, and if we were to normalize the data so that every metric could exist on a standardized scale, it would've been a lot easier to make predictions. If we were given the opportunity to extend this research further, we would have to look into more factors affecting a song's popularity. These factors range from the presence of social media platforms in addition to the size of a particular artist's following. Adding more model features from other platforms and cross-referencing the ranking of songs on those platforms could improve the model and more additional complexity. Additionally, we didn't account for the genre. Many genres have different baseline sounds unique to that one genre, and we believe that this may have caused some discrepancies between the predicted and actual rankings of each song.

This analysis has many possible real-world applications, with some identifying general music trends, allowing artists to increase popularity. Additionally, since the analysis can provide artists with information on what a particular demographic is responding well to, an artist can tailor their music to reflect the taste of that demographic, increasing popularity within that region. Finally, with some modification, the machine learning models created can be used within a recommendation engine to provide tangential or similar sounding songs to what a demographic is listening to.

Citations:

charlie86. (n.d.). *Charlie86/Spotifyr: R wrapper for Spotify's web api*. GitHub. Retrieved May 2, 2022, from https://github.com/charlie86/spotifyr

Financials. Spotify. (n.d.). Retrieved May 2, 2022, from https://investors.spotify.com/financials/default.aspx

Götting, M. C. (2021, January 8). *Music listening habits in the U.S. by age 2019*. Statista. Retrieved May 2, 2022, from https://www.statista.com/statistics/749666/music-listening-habits-age-usa/

MODELMETRICS documentation. ModelMetrics documentation. (n.d.). Retrieved May 2, 2022, from https://rdrr.io/cran/ModelMetrics/man/

Ramchandani, P. (2021, April 10). *Random forests and the bias-variance tradeoff*. Medium. Retrieved May 2, 2022, from https://towardsdatascience.com/random-forests-and-the-bias-variance-tradeoff-3b77fee339b4

Spotify charts - Spotify charts are made by fans. Spotify Charts - Spotify Charts are made by fans. (n.d.). Retrieved May 2, 2022, from https://charts.spotify.com/charts/view/citypulsetrack-denver-weekly/latest

Spotify Research. (n.d.). Retrieved May 2, 2022, from https://research.atspotify.com/

Spotify Web API. Spotify for Developers. (n.d.). Retrieved May 2, 2022, from https://developer.spotify.com/documentation/web-api/

Visualize correlation matrix using correlogram. STHDA. (n.d.). Retrieved May 2, 2022, from http://www.sthda.com/english/wiki/visualize-correlation-matrix-using-correlogram