ISyE 6740 - Project Proposal
Song Popularity Prediction Using Multimodal Data

Andrew Taylor - GTID 902697379

1 Motivation & Problem Statement

1.1 General Motivation

The reasons behind why people listen to certain songs can be difficult to nail down. It could be the way the song sounds - are popular songs generally more upbeat? Are they more "danceable" than others? Are they in major or minor keys? Common 'singable' keys like C, D, and G? It could be things having nothing to do with the audio itself - is a song's popularity driven in part by the previous success of its artist? The number of followers that artist has already accumulated?

1.2 Problem Statement

Given data from Spotify about a song's audio qualities, metadata from Spotify and Musicbrainz about the song's artist, and data from last.fm's API that provide streaming count, we tested and evaluated various regression measures to answer two questions:

1. How well can we predict a song's streaming popularity using just the audio analysis of the song and information about the artist?

2. What is the relationship between a song's audio features / artist information and its play count? (Are the relationships statistically significant, and what do they imply?)

1.3 Context & Novelty

This topic would be useful for a number of reasons to someone in the music industry, and as such has been explored to some length already. If certain characteristics make a song more or less prone to commercial success, then future song producers, writers, and musicians might be better informed to write successful music.

Preliminary investigation of articles written on this topic seem to reveal a few things. Past analysis appears to primarily focus on classification problems that predict whether a song will be a hit oo not, or predict popularity on a Low-Medium-High scale. In addition, past attempts have explored audio features, text features of lyrics, and the some of the same Spotify API metadata we will be using.

What we hope to bring new to the table is the perspective of a multimodal dataset (as opposed to only looking at waveform analysis, lyrics, or audio information), as well as a different dependent variable. Most of the existing studies use either Billboard's top 100/200 to classify a song as a

'hit', or use Spotify's opaquely described and point-in-time 0-100 'popularity' measure. Instead, we quiered last.fm's API with a track's Musicbrainz ID to pull a play count measure from last.fm.

2 Dataset

The data used for this project was sourced from multiple online resources, and the field list is diagrammed below.

Field	Data Type	Role	Source
Play Count	Float, (Unbounded)	Dependent Variable	last.fm API
Danceability	Float (0-100)	Feature (Continuous)	Spotify API
Acousticness	Float (0-100)	Feature (Continuous)	Spotify API
Liveness	Float (0-100)	Feature (Continuous)	Spotify API
Valence	Float (0-100)	Feature (Continuous)	Spotify API
Loudness	Float (0-100)	Feature (Continuous)	Spotify API
Energy	Float (0-100)	Feature (Continuous)	Spotify API
Instrumentalness	Float (0-100)	Feature (Continuous)	Spotify API
Tempo	Float	Feature (Continuous)	Spotify API
Key	Integer	Feature (Categorical)	Spotify API
Time Signature	Integer	Feature (Categorical)	Spotify API
Modality	Integer	Feature (Binary)	Spotify API
Artist Num. Followers	Float (Unbounded)	Feature (Continous)	Spotify API
Artist Popularity	Float (0-100)	Feature (Continous)	Spotify API

Figure 1: Initial Feature Set

Some data sources were easier to access and extract from than others. We began by downloading a snapshot of the Musicbrainz database, which included 36M tracks in the 'tracks' table, but required some querying across 8 different tables to get to the data we needed. This included limiting our results to songs which had a stored International Standard Recording Code (which we can use in Spotify's API), and filtering to recordings which were released in 2019 or later to concentrate the study. This left us with 8M records, from which 10% were sampled to be queried against Spotify and last.fm's API. This sample size was based on daily rate limits set by Spotify (which, unfortunately, are not stated in their documentation but were learned the hard way).

To go from a list of ISRCs to individual track records with artist information and audio features, a number of Spotify API calls were required. First, we used the 'search' API to search the ISRC to get the spotify song Uniform Resource Identifier (URI), during which 1.7% of ISRCs could not be found on Spotify, although no discernable pattern could be found. Then, the track.getInfo API call was used to get the artist URI. The artist.getInfo API call was used to pull artist information, and finally, the track.getAudioFeatures call was used to pull track audio analysis.

In the last.fm environment, unfortunately, the track Musicbrainz IDs were not stored, although they are stated to be in the documentation. Instead, we used the Musicbrainz IDs for the release (album) to get the track list, during which call 15% of MBIDs were not able to be matched. Then used a separate API all to get each track's play count, during which call 20% of track names and artist names were not able to matched. In both cases, no discernable pattern of loss was apparent.

Finally, the single dataset was assembled using a series of joins, as well as fuzzy matching to match track names within an album. The final dataset consisted of $\tilde{3}$0,000 records, each of which corresponded with a song and had complete data across the sources discussed.

Additional materials are in the appendix - the initial plan for our prep process is in Figure 5, and Figure 6 shows the full schema of the Musicbrainz database.

3 Methodology & Approach

3.1 Overview

We ultimately broke this project out into five phases - data collection, data cleansing & preparation, exploratory data analysis, model evaluation for predictive value, and model evaluation for feature insights.

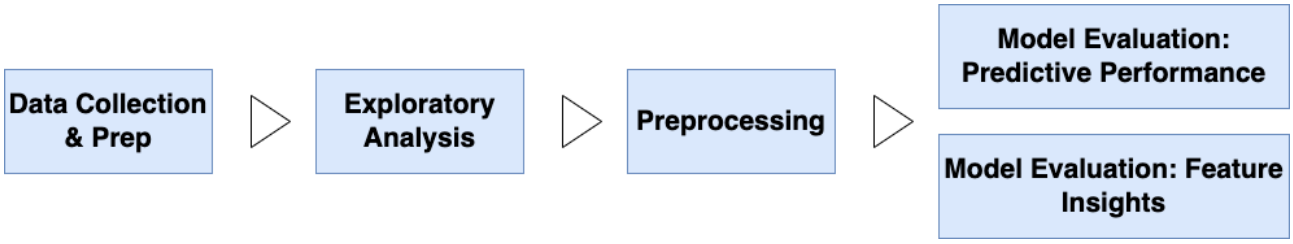

Figure 2: Project Approach

3.2 Data Collection

In the Data Collection phase, we went through the steps described in the 'Dataset' section above - please review that section for details.

3.3 Exploratory Analysis

In the Exploratory Analysis phase, we conducted a series of univariate and multivariate summaries to understand each feature, its distribution, and their relationships to each other and the dependent variable.

First, we performed univariate analysis (histograms, box plots, count plots, and single variate statistics) to better understand each feature, as well as the dependent variable. The observations can be found below.

Play Count

- Highly right skewed but fairly normally distributed on a log scale
- We may want to consider a log or Box-Cox transformation of the response variable when evaluating linear models

Danceability

- Evenly distributed between 0 and 1

Energy

- Slightly left-skewed, but no apparent outliers

Key

- Somewhat evenly distributed across 11 keys
- Key value of 3 (key of E flat) is relatively unpopular
- Key values of 0, 1, 2, 7, 9 highly popular (keys of C, C#, D, G, and A)

Loudness

- Slightly left-skewed, with a center and large concentration around -10 db

Mode

- Relatively evenly distributed between 0 (minor key) and 1 (major key)

Speechiness

- Highly right-skewed
- Could consider removing outliers or not considering this variable

Acousticness

- Right-skewed
- Could conside consider removing outliers or not considering this variable

Instrumentalness

- Right-skewed
- Appear to have categories of ¡.5 (not instrumental) or ¿0.8 (instrumental)

Liveness

- Right-skewed
- Appears to have a bump around .3-.4 - unsure why (maybe simiulated audience noise?)

Valence

- Slightly right-skewed, with a slight majority having less than 0.5 value (sound "sad")

Tempo

- Fairly normally distributed, cenered around 120 beats per minute

Duration (ms)

- Highly right-skewed

Time Signature

- Vast majority of songs are in 4/4, with a few in 3/4 and very few in any other signature

Artist Popularity

- Relatively normally distributed

Artist Followers

- Relatively normally distributed on a log scale - right-skewed, otherwise

We see that a number of predictors show skewness. However, in the initial runs of the models, we do not transform these - given the relatively large dataset (¿10,000), the models should be relatively robust.

In multivariate analysis, we saw that most variables are fairly uncorrelated (—corr(x1, x2— ¡ 0.2—) with the exception of acousticness and loudness/energy, which are more highly negatively correlated. We will leave them all in, but may consider variations of model runs that remove acousticness from the model.

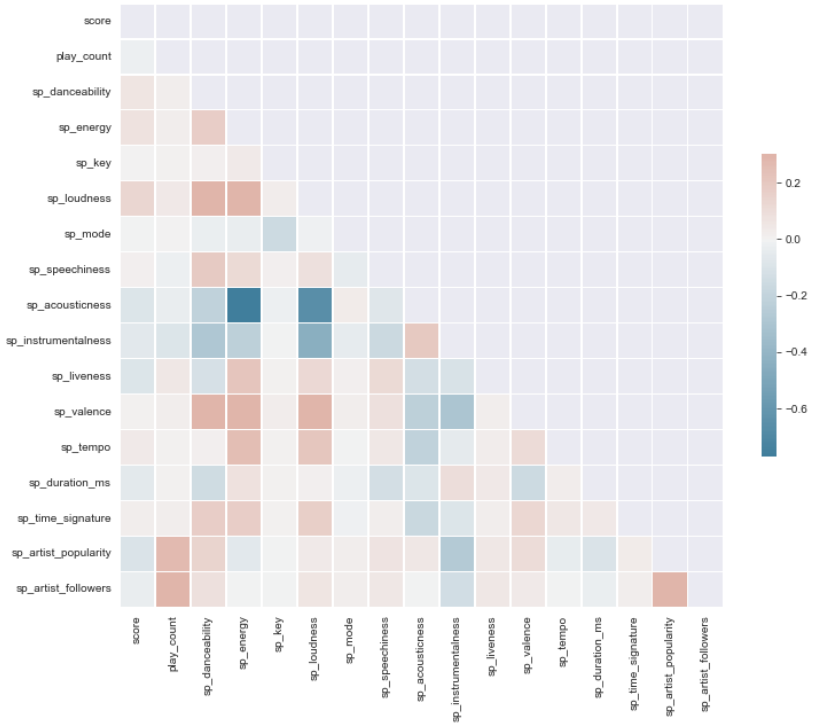

Figure 3: Correlation Plot

In creating scatterplots of each variable against the dependent variable (Figure 4), there do appear to be a few trends:

- Energy appears to have some positive relation with play count
- Artist popularity appears to have the some positive relation to play count

- Artist followers appears to have some postive relation to play count
- Mode shows that minor songs tend to have higher play count than major
- Time signature shows that songs in the 4/4 time signature tend to have higher play count than other time signatures

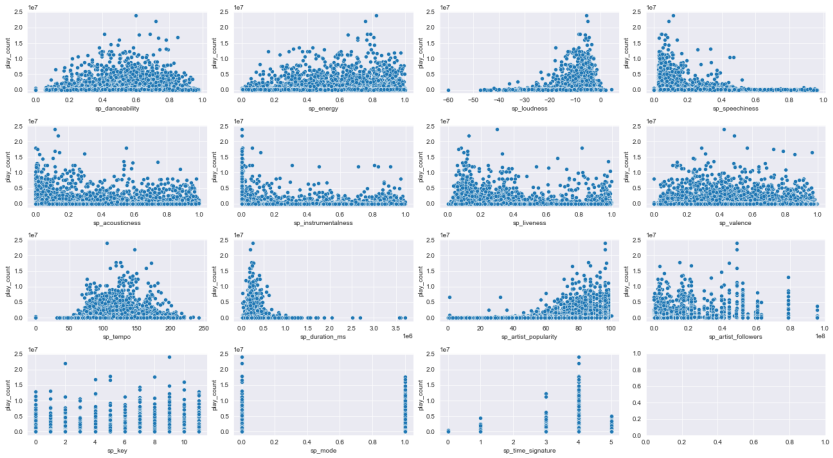

Figure 4: Features vs Play Count

3.4 Preprocessing

During data preprocessing, we primarily needed a way to handle our categorical variables, then we needed to split the model into a test and training set. First, we one-hot encoded the categorical variables (key, mode, and time signature). One-hot encoding was selected over Ordinal Encoding, since none of the three variables were ordinal in nature. Then, 20% of the data was set aside for testing, with the remaining 80% in a training-validation set.

3.5 Model Evaluation - Overall

We split our model evaluation into two parts based on our two questions. First, we want to see if, based solely on these features, we can generate a regression model which predicts play count reasonably well. Second, we want to again use regression to better understand which features might have a significant relationship with play count. We do not believe that the set of models analyzed for both questions needed to be the same, since there was are many valid and recently popiular regression models (e.g. gradient boosting and random forest regressors) which may not be informative enough at the feature-level.

Our specific plan was to pick a subset of model families, some of which were interpretable at the feature-level (i.e. provided coefficients) and some of which were not. Then, on our training-validation dataset (the 80% sample), we would use nested cross-validation to tune the model hyperparameters in the inner loop, and get a validation score (R^2 score, MSE, and MAE) for model selection in the outer loop.

From there, two things happen in parellel. We would select the best overall model to be dubbed the 'most predictive' and help answer our first question, and we would select the best model of

the 'feature interpretable' models to help answer our second question. Then, for the predictive model, we would analyze the results and hyperparameters, train the selected model on the full training-validation set, and test the performance on the 20% test set to get a final score. For the feature-interpretable model, we would analyze p-values and coefficients to understand relationships between features and the play count.

This was all performed using various scikit-learn libraries for model selection, evaluation, and creation. In addition, the statsmodels library was used to dive deeper into the selected model for feature interpretability, as it has stronger p-value and stats reporting capabilities than scikit-learn.

We selected 3 folds on the inner CV loop where it was needed to select hyperparameters, and selected 10 folds on the outer loop. Choosing 10 folds on the outer loop also allowed us to examine the hyperparameters selected for the estimator in each loop and get an idea of how robust the hyperparameters were to randomness.

We ultimately chose to evaluate the following model types:

Model	**Type**	**Hyperparameters Tuned**
Linear Regression	Feature-Interpretable	None
Lasso Regression	Feature-Interpretable	alpha (regularization constant)
Ridge Regression	Feature-Interpretable	alpha (regularization constant)
Random Forest Regression	Feature-Opaque	number of trees, maximum tree depth, maximum features
Gradient Boosting Regression	Feature-Opaque	number of boosting iterations maximum tree depth, learning rate/shrinkage, subsample %, maximum features
Light Gradient Boosting Regression	Feature-Opaque	number of boosting iterations, maximum tree depth, learning rate/shrinkage

3.6 Model Evaluation - Model Comparison & Hyperparameter Tuning

In the first step, nested cross-validation allows us to tune the models and select two models - one for each key question. The results were as shown below.

Model Selection Results - Scored on the Training-Validation Set

Model	Type	R^2 Score	MAE	MSE
Linear Regression	Feature-Interpretable	0.1816	-273,220	-6.67E+11
Lasso Regression	Feature-Interpretable	0.1818	-270,388	-6.67E+11
Ridge Regression	Feature-Interpretable	0.1817	-272,272	-6.67E+11
Random Forest Regression	Feature-Opaque	0.3023	-225,363	-5.68E+11
Gradient Boosting Regression	Feature-Opaque	0.3537	-213,749	-5.22E+11
Light Gradient Boosting Regression	Feature-Opaque	0.3129	-222,858	-5.57E+11

Based on all three error metrics, we noted that Gradient Boosting Regression was providing the strongest, albeit still moderately weak, accuracy scores, and was selected as the best predictive model. In the next step, we examined more closely tuning parameters to understand how to further improve this model. Overall, we also noticed that our nonlinear models (Random Forest, Gradient

Boosting Trees, and LightGBT) performed fairly significantly better than our linear models. This seemed to make sense, as we have a number of categorical variables in our data that may provide ways to explore the data in a nonlinear fashion, as opposed to the linear models (linear regresion, lasso, and ridge). We also did not perform any polynomial or other feature transformations that might make the linear models more performant.

Based on all error metrics, we saw that Lasso was providing the strongest, albeit weak, accuracy scores of the feature-interpretable models, and was selected as the best feature-interpretable model. We also noted that each iteration of Lasso was typically removing 8-11 variables from the model. In the next step, we examined the feature p-values in order to udnerstand whether feature insights were relevant, even if the predictive power was limited.

3.7 Predictive Power Evaluation - Gradient Boosting Regression

In the initial model selection phase, a grid search was performed on the gradient boosting model's learning rate (which controls shrinkage, or the influence of previous models on each boosting iteration's update), max depth (which controls the distance from the root node to the furthest leaf along an individual tree in a single estimator), max_features (which controls the number of features a single split can consider), and n_estimators (the number of boosting iterations done and subsequently weak learners used in the final prediction). We opted to use max_depth as the initial control for tree depth (and consequently overfitting), then use the results to determine if other parameters needed to be tuned (min_samples_split, max_sample_leaf). Our learnings from those tuned hyperparameters are below. We performed an additional round of hyperparameter tuning on the training-validation set after selecting gradient boosting to further optimize the hyperparameters before evaluating performance on the testing set.

Gradient Boosting Hyperparameter Learnings

- **learning_rate -** In a search range of 0.01 to 0.04, 0.04 was selected in all 10 internal cross-validation folds. Higher learning rates were explored in the fine-tuning round, although many sources stated that lower rates often provide better results (albeit at the cost of increased computation)
- **max_depth -** In a search range of 2 to 20, a maximum depth to 4 was selected in all but one round, during which a lower max_features value was selected to avoid overfit. This value was held constant at 4 in the fine-tuning round
- **max_features -** Two values were tested - 'auto', which allows splits to use all features, and 'sqrt', which allows splits to consider at most sqrt of the total number of features. 'Auto' was selected in all but one round, and was used during the fine-tuning round.
- **n_estimators -** In a search range of 50 to 500, 500 was selected in 8 rounds and 250 was selected in 2 rounds. A value of 500 was chosen for the fine-tuning round. It's not surprising that more estimators provide accuracy improvements, as many sources state that boosting is fairly robust to overfit with more estimators.
- **subsample -** In a search range of 0 to 1, 0.9 was selected 9 of 10 times, and was held constant in the fine-tuning round.

Next, we fine-tuned the model, evaluating a larger range of learning rates (0.4 to 1). Because the max_depth was consistently chosen at 4 (relatively shallow), we did not take any further measures

to limit overfitting. We ran this evaluation through the nested cross-validation process (with the same random seeds) again to ensure that it didn't degrade the performance, and found that the performance was identical with a learning rate of 0.4 selected.

Finally, we re-trained the model with fixed hyperparameters on the full training-validation set, and tested the performance on the test set. When we did that, the results were as follows:

Final Gradient Boosting Results - Scored on the Test Set

Model	Type	R^2 Score	MAE	MSE
Gradient Boosting Regression	Feature-Opaque	0.3573	-212,790	-5.18E+11

The performance continued to be a challenge, with moderate R^2. In the Conclusions and Next Steps sections, we discuss the implications of this and some possible ways to explore improving the model beyond the multiple rounds of hyperparameter tuning conducted in this project.

3.8 Feature Insight Evaluation - Lasso

In parallel, after selecting Lasso as our method for understanding feature relationships, we needed to select and train a final model and use it to both select features and understand their significance / relationship to play count.

For Lasso, the only hyperparameter that was searched was the alpha-value (which controls the magnitude of the L1-regularization - the higher the alpha-value, the more penalty given to having a higher number of features). In the initial rounds of cross-validation for model selection, we saw alpha values in the range of 600-800 frequently selected, although the number varied fairly frequently. This might imply that randomness in the data had moderate impact on the importance of certain variables, leading to different alpha values winning out in different folds / loops of cross-validation.

In order to move forward with a specific alpha value, the inner loop was run a final time on the full training-validation dataset. In this run, an alpha-value of 708 drove the best performance. This alpha value was then used in the statsmodels implementation of Lasso regression (again on the training-validation dataset) in order to review the statistical summary of the coefficients (shown below). This model had another weak R^2 score of 0.185, but the p-values for many values were less than 0.01, showing that even though a linear Lasso model may not have much predictive power, many coefficients have a strong relationship with the play count. *Note:* In the table below, a feature is considered 'significant' if it had a p-value below 0.01, 'slightly significant' if it had a p-value below 0.05, and 'not significant' otherwise.

Feature	p-value	Significance	Coefficient
const	0.3198	not significant	-187,558
sp_danceability	0.0012	significant	-112,274
sp_energy	0.1339	not significant	65,641
sp_loudness	0.05	not significant	-3,755
sp_speechiness	0.00	significant	-392,859
sp_acousticness	0.0004	significant	-91,933
sp_instrumentalness	0.0038	significant	-54,228
sp_liveness	0.00	significant	144,098
sp_valence	NaN	not used	0
sp_tempo	0.8063	not significant	47
sp_duration_ms	0.0161	slightly significant	0
sp_artist_popularity	0.00	significant	4,414
sp_artist_followers	0.00	significant	0
sp_mode	0.1608	not significant	-16,381
sp_time_signature_0	NaN	not used	0
sp_time_signature_1	0.8498	not significant	36,252
sp_time_signature_3	0.725	not significant	65,187
sp_time_signature_4	0.6779	not significant	76,739
sp_time_signature_5	0.8762	not significant	29,352
sp_key_0	0.4054	not significant	15,095
sp_key_1	0.0017	significant	-61,711
sp_key_2	NaN	not used	0
sp_key_3	NaN	not used	0
sp_key_4	NaN	not used	0
sp_key_5	0.025	slightly significant	-47,389
sp_key_6	NaN	not used	0
sp_key_7	NaN	not used	0
sp_key_8	0.0642	not significant	-44,361
sp_key_9	0.252	not significant	21,840
sp_key_10	0.6968	not significant	-9,194
sp_key_11	NaN	not used	0

Feature Observations

- **Ignored features** - Valence, whether or not a song has no discernable time signature, and a number of keys (D, Eb, E, F#, and G) did not have significant enough impact to be kept by the Lasso algorithm's regularization. This makes some sense, given that many songs (popular and unpopular) are written in those keys
- **Significant Feature - Artist Followers** - The number of followers an artist has was highly significant, with a positive coefficient indiciating that each 1 additional follower may lead to as much as 0.04 additional listeners to a particular song
- **Significant Feature - Artist Popularity** - This similarly tells us that Spotify's measure of artist popularity (vaguely described as calculated from the popularity of the artists' tracks)

- **Significant Feature - Speechiness** - This showed a negative relationship, meaning that the more talking a track contains, the fewer plays it gets
- **Significant Feature - Liveness** - Interestingly, this showed a positive relationship, meaning the more live a track was, the more popular it was
- **Significant Feature - Acousticness** - This showed a negative relationship, meaning that the more acoustic a track is, the fewer plays it gets
- **Significant Feature - Danceability** - This showed a negative relationship, which was surprising - one might think that a more 'danceable' track is more likely to be popular
- **Significant Feature - Key of C#** - This showed a negative relationship, which makes some sense. This was the only significant feature related to keys - many others (especially popular keys) were excluded from the model. It seems that this is a less popular key to compose in (fewer songs were in C# than many other keys), and led to lower play count
- **Significant Feature - Instrumentalness** - This showed a negative relationship, meaning that the more interumental a track is, the fewer plays it gets. This might make some sense - it does seem that many people enjoy tracks with lyrics
- **Slightly Significant Feature - Duration** - This showed a positive relationship, although very slight
- **Slightly Significant Feature - Key of F** - This showed a negative relationship (interestingly) - perhaps for the same reason as the Key of C#

4 Conclusions

In this study, we've determined that there are some features of music, both artist information and audio analysis, that have a significant relationship with their play count. However, developing a strong predictive model using just artist information and audio analysis proved difficult, with the most performant models only providing moderate to weak predictive power.

5 Next Steps

As we think about ways this could be improved, there are a number of things that come to mind. We've broken these down into categories.

Ways to Increase Predictive Performance:

- Increase the sample size from the dataset (paying for higher rate limit from Spotify's API / taking multiple days-weeks to collect data)
- Add additional features that may be more correlated with listens, such as a song's popularity on Tiktok
- Consider partioning data (e.g. by genre or world region) to create a more specific model

Ways to Improve Predictive Model Train Speed:

- Further explore increasing LightGBM model accuracy as a potential substitute
- Analyze the n_predictors hyperparameter further to determine optimal number beyond which performance increases are marginal

Ways to Improve Feature Significance Models:

- Add hypothetical interaction features to model to understand if they provide additional insights (e.g. combining model and key to understand if A-minor keys have a significantly different relationship with the response than the separate A key and minor mode variables).
- Consider a log transformation of the response variable

6 Appendix

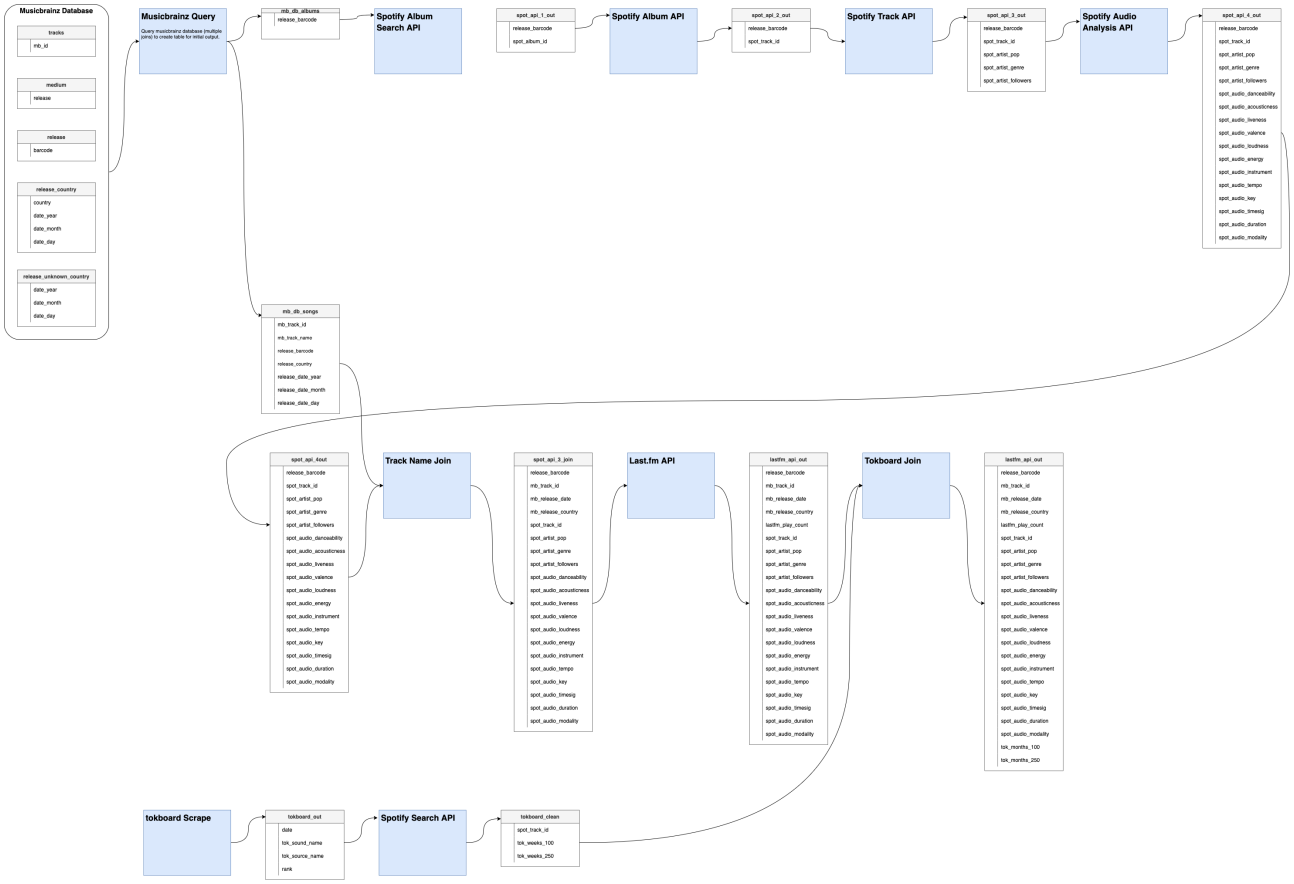

Figure 5: Initial Data Prep Diagram

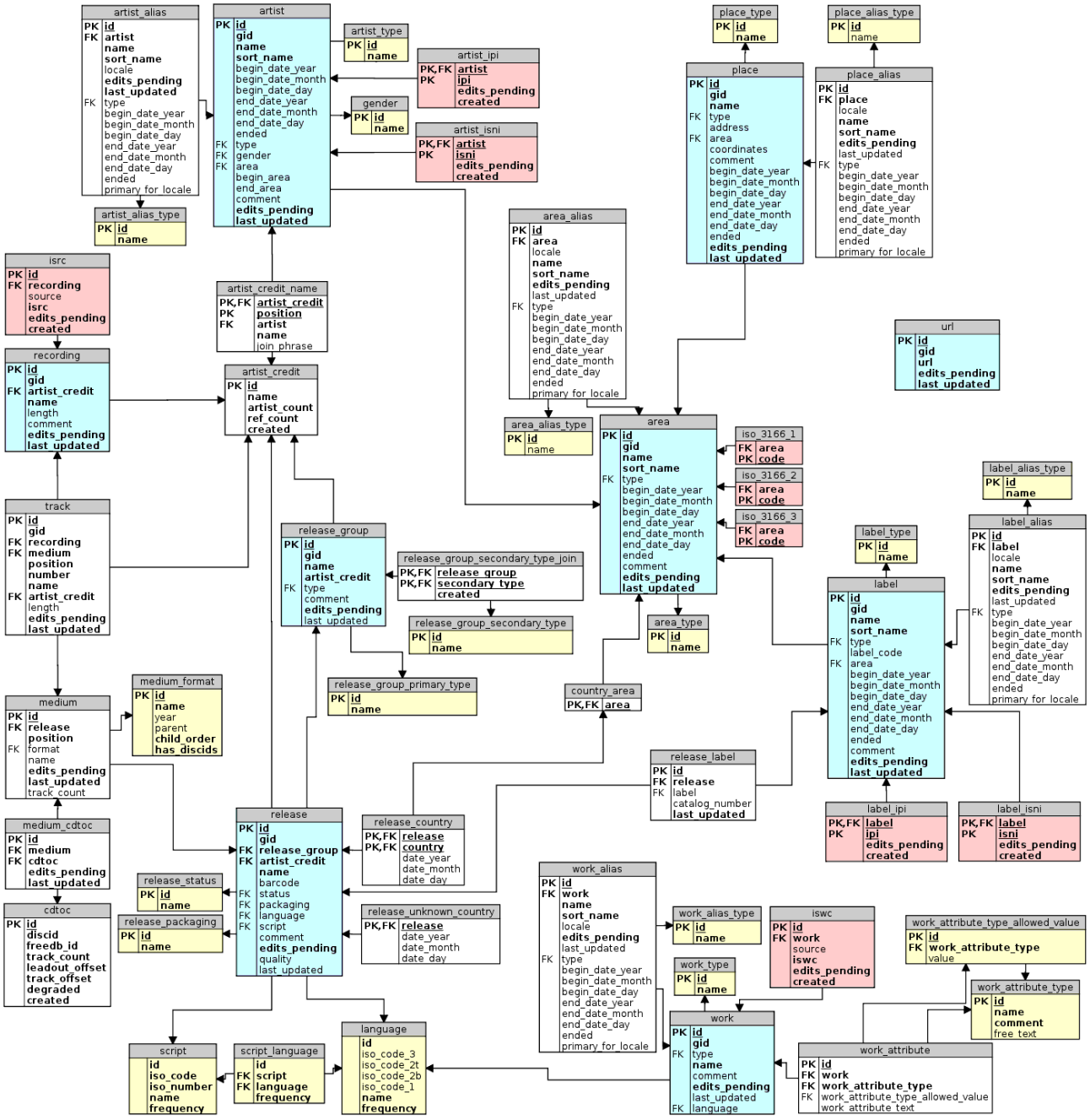

Figure 6: Musicbrainz Schema