# Spotify ML Project: Pre-Analysis Plan

## Introduction and Analysis Questions

As previously mentioned in the Wrangling/EDA portion of this project, our study is centered around Spotify data. More specifically, we are interested in understanding more about the popularity of Spotify songs. As a reminder, observations within this study are individual songs that are represented by various features provided by the Spotify API along with popularity scores. Each observation includes the variables: danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo and duration. Each song represents an observation with its respective feature values and a popularity score. With this being said, our primary goal in this analysis is to better understand these questions:

1. **Which song characteristics are most useful in predicting the popularity of a song? In other words, what characteristics make a song popular?**
2. **How well can we predict song popularity for future songs, given the variables provided by the Spotify API?**

To answer these questions, we will be performing supervised learning for regression, as our data is labeled and the response variable is quantitative. To perform this regression analysis, we will be using a Random Forest Model. We will make predictions using a Random Forest Model, assess the accuracy of our predictions, and then analyze the feature importance of the model to see which variables are most important in predicting Popularity.

## Model Selection and Considerations

The Random Forest Model was chosen because it performs well for predictors that may have some multicollinearity. As previously mentioned in Part 1 of this project, some of the variables within our data are likely to give redundant information, as they are very similar. This idea was proven by a correlation matrix showing high correlations between certain predictors. Additionally, the Random Forest Model does not need the response variable to be normally distributed. Based on our EDA, we found that the Popularity variable is almost bimodal, as so many songs had a ranking between 0 and 10. For this reason, the normality assumption may not be met, so the Random Forest Model is a safe choice.

One consideration before fitting the Random Forest Model is the parameters that will be used. Some key parameters for a random forest model include: number of trees, maximum tree depth, and minimum samples per leaf. To find the optimal parameters for our model, we will first start with the default parameters. We will then use cross-validation to assess model performance on the training data, looking for metrics like accuracy and RMSE. Next, we will check for underfitting/overfitting of the model using the testing data. Depending on these results, we will adjust the parameters accordingly, until we come up with a correctly fitted final model.

Before performing this analysis, we will split the data into a training set and a testing set. The training data will be 80% of the original data, while the testing data will be 20% of the data. Next, we will use bootstrapping to create multiple subsets of the original training data. Each subset is created by randomly sampling with replacement from the training dataset. We will then plug the training data into the Random Forest model, to train the model. During this process, a decision tree is built for each subset of the training data. Once all trees are built, predictions will be made by taking the average of all decision tree predictions. After the model has been trained, we will assess the accuracy of the model using data that has never been seen by the model (the testing data). This will be done so that we can accurately see how well our model predicts popularity, as well as check for any overfitting/underfitting. If the fit of the model is incorrect, we may need to reassess the parameters chosen for the Random Forest Model.

## Feature Engineering

To prepare our dataset, we will implement several feature engineering techniques aimed at enhancing the predictive power and usability of the data. First, we will handle categorical variables such as 'mode,' `key,' `year,' and `month.' The `mode` variable, which has already been relabeled as "Major" or "Minor," will be one-hot encoded to convert these categories into binary indicators, allowing the model to incorporate modality, the type of musical scale used in a piece which gives the music its particular tonal quality or "mood,"  without introducing numerical bias. Similarly, the `key` variable, which has been mapped to note names, and the `month` and `year` variables will undergo one-hot encoding, allowing each distinct category to contribute separately without introducing multicollinearity.

In addition, we'll address potential multicollinearity among numeric predictors, which include features like `danceability,` `energy,` `loudness,` `speechiness,` `acousticness,` `instrumentalness,` `liveness,` `valence,` and `tempo.` Since some of these variables (such as `instrumentalness` and `speechiness`) may provide overlapping information, we can calculate the correlation between the variables to identify highly correlated pairs. For variables showing strong correlations, we may apply Principal Component Analysis (PCA) to reduce redundancy. PCA will transform a set of correlated variables into a smaller number of uncorrelated variables. Each principal component is a linear combination of the original variables and is designed to capture as much of the data's variance as possible in fewer dimensions. For example, variables like `danceability`, `energy`, and `loudness` are often correlated in music data, so PCA would combine these into new components that represent the core information without redundancy. This reduction in multicollinearity can make the model more robust by minimizing redundant information, while the reduced feature set improves computational efficiency.

## Assessing the Model

After fitting the model, we will need to check the accuracy of our model in order to interpret how well our model works for predicting the popularity of a song. To do this, we will first calculate the R^2 value, which will tell us what proportion of the variance of the independent variable is explained by our model. A high R^2 value indicates that the model is a good fit for prediction. Generally, we would like to have a model with an R^2 value of at least over 0.5. An R^2 of 0.5 indicates that half of the variance is explained by the model, which is not extremely strong, but is typically considered decent. As this model is on Spotify data and is not high-stakes, this would be an okay R^2 value to have, although we hope for an R^2 value of closer to 0.8.

We will also calculate the R^2 value for the training set, which is the dataset that we initially trained our model on. Ideally, the R^2 value should be approximately equal for the training set and for the test set, because this means there is no overfitting or underfitting occurring. If the R^2 value is much higher on the training set than the test set, we know that our model is overfitting the data, which means we likely have too many predictors which make our model no longer generalizable.

To calculate another measure of accuracy, we will calculate the root mean squared error (RMSE) of the predictions we made on the test set. Since this value represents the error we made in our predictions, a higher RMSE indicates that our model is not a good fit as it currently stands.

In the case that our R^2 value is lower than 0.5 and/or our RMSE value is alarmingly high for this dependent variable's scale, we will revisit the parameters we had initially set. We will initially set 'n_estimators' equal to 100, and 'max_depth' equal to None. This means that we will have 100 decision trees averaged to get our Random Forest Model, and the trees have no maximum depth. However, 'max_depth' set lower can help combat overfitting, if that is a problem we find.

To present our results from this model building process, we will show our final model on the training set, the predictions on the test set, the R^2 values for each set, and the root mean squared error of the predictions on the test set. This will give us a picture of how successful our model is at predicting the popularity of songs, which can allow us to determine if this model would be useful to our stakeholders.

## Possible Challenges

With this model, we anticipate running into challenges that stem from the fitting of the model. Random Forest models tend to have a problem with overfitting on complicated datasets. Another source of fitting problems could stem from hyperparameter sensitivity of the tree, like maximum depth and number of observations per branch. This will be assessed thoroughly with RMSE calculations and adjustments made to the model to maximize $R^2$. If our model does perform poorly due to parameter selection, this may help us learn more about how to tweak the parameters of a Random Forest Model to correctly account for overfitting/underfitting.

Another challenge we expect with this model is the feature importance score calculations. The feature importance poses a higher threat for data sets that tend to have variables or parameters with coinciding or codependent information. With the analysis for multicollinearity of the parameters, this challenge should be significantly reduced, as the collinear parameters will be assessed by principal component analysis. There could also be a challenge with extrapolation of the model to apply to other song data.

Although we have properly cleaned and explored our data from the Spotify API, the data is still limited, as it only includes songs that are featured on Spotify released between 2004-2019. This could cause an imbalance in the data used to train and test the model, as some older songs (before 2004) and new songs may have parameters that are technically outside of the model's range. By ensuring a balanced train and test data split, this model will allow us to assess the importance of the parameters in determining the popularity of a song.