Stage 3 report

Introduction

The best example to illustrate the commercialization of the arts is today's music industry. Being a multi-billion dollar industry, artists and labels have to work on finding a balance between making music that appeals to their fans and fulfills their artistic expression, while also makes money. In the digital landscape, the number of plays or streams a song gets is indicative of it's financial success. In other words, the more popular a song becomes, the more money it brings in. If this really is the case, then being able to idenfity how popular a song can become before it is even released would be of great interest to labels and record companies. So, this naturally raises the question that we will be focusing on in this report:

Given data on attributes of a song, can we predict how popular it will become?

Dataset

To answer this question, our group will be working with data on different audio-based features of a song to predict it's popularity when released on a music streaming platform. The dataset we will be using is from the Tidy Tuesday dataset collection [https://github.com/rfordatascience/tidytuesday/blob/master/data/2021/2021-09-14/readme.md]. We have cleaned and combined the `audio_features.csv` and `billboard.csv` datasets provided to obtain the dataset we will use for the rest of our analysis. The raw data files have 32 variables, of which we will be using 21 relevant variables, removing the variables that don't affect our dataset, such as URLs and id's for the songs. We also narrowed the billboard data by selecting only the rows which a song peaks on the chart, and ignoring other weeks, then removing the week specific variables. The full transformation can be found in the `Data_Transformation_script_Billboard.R` script. The remaining data represents different features of a song such as its key, tempo, loudness, danceability, etc. as well as metrics reflecting its popularity on billboard and spotify charts respectively.

Objective

Our goal with this project/report is to create a regression model to predict how "popular" a song may be based on it's audio-based features. This means we will be looking at each song independent of cultural trends, the associated artist's popularity, and other external influences. We will be looking into ideas such as what features popular songs have in common, if any specific feature is highly linked to a song performing well or being well recieved, and whether we can identify any subgroups with their own unique trends.

Motivation

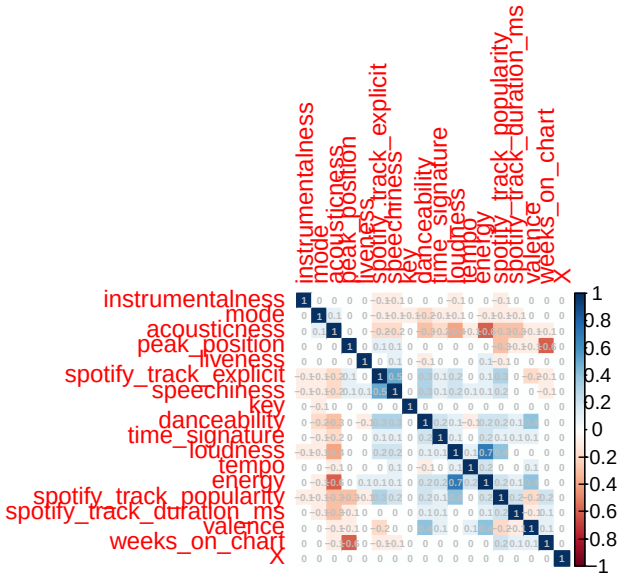
We as a group agreed to working with this dataset and on this question because we are all avid music lovers. With varied interest and tastes amongst all of us, we thought it would be interesting to see how an emperical analysis of what is or isn't "popular music" would compare to our own personal preferences, and how much each of us may agree or disagree with the findings.

Exploratory Data Analysis

Quick briefing of selection of response variable

We first cleaned our dataset by removing obviously non-useful variables like names. (However, while we recognize that different genres may have differing levels of popularity, for simplicity we will leave out this variable as the "genre" of certain songs may be ambiguous and there are too many categories to keep track of.)

Next we make a correlation plot to have a broad overview of what variables may be connected to our response as well as with each other:

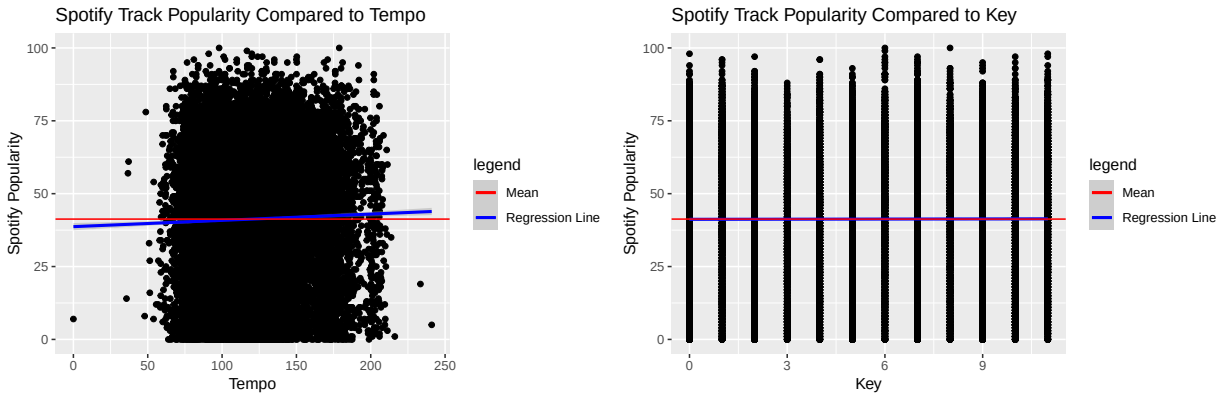


We had briefly explained in our stage 2 report that we originally chose `peak_performance` as our response variable, but with its noticeably low correlations with most of the predictors, we switched our response to `spotify_track_popularity`.

Exploration of Spotify Track Popularity on Billboard as Response Variable

`spotify_track_popularity` is correllated with almost all parameters in our data set. This means we are much more likely to be able to actually predict it's popularity based on the values of the other parameters, which will preform better than `peak_performance` did.

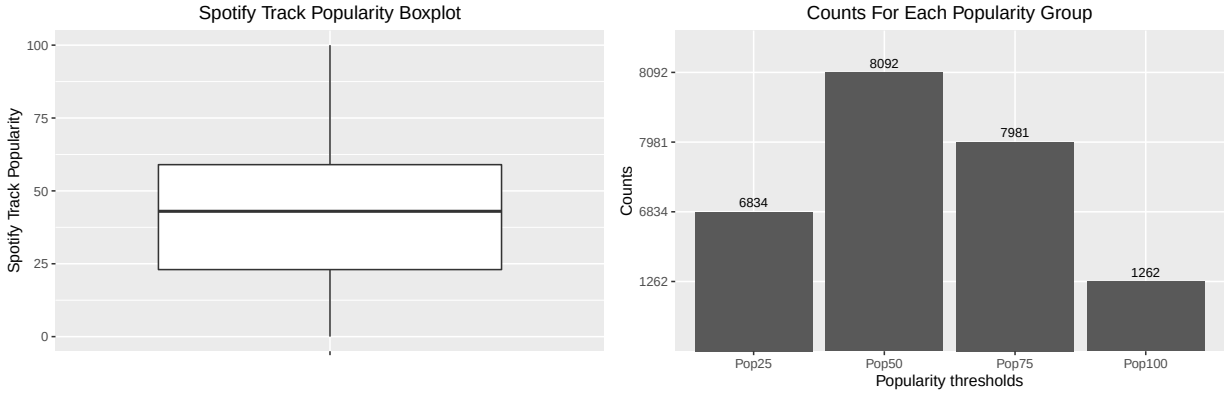
Here we see the two least correlated variables with respect to `spotify_track_popularity`:



These two variables have been explained in greater detail in the earlier report, but are still quite relevant to mention as in our data analysis in training and testing models. These two were filtered out due to 0 correlation with the response `spotify_track_popularity` and cut down total number of predictors.

As we've seen, all these variables in the correlation matrix do have more correlation in some form in tandem with `spotify_track_popularity`, hence it should be a better response variable than `peak_performance`, so we will use this as our response variable.

Further Exploration Of Data




```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   23.00   43.00   41.28   59.00  100.00
```

Spotify popularity ranges from 0 to 100. The boxplot alongside the five number summary helps us understand that the median popularity is 43. This suggests that a majority of the songs are ranked lower in popularity, hence the higher popularity ranks are coveted.

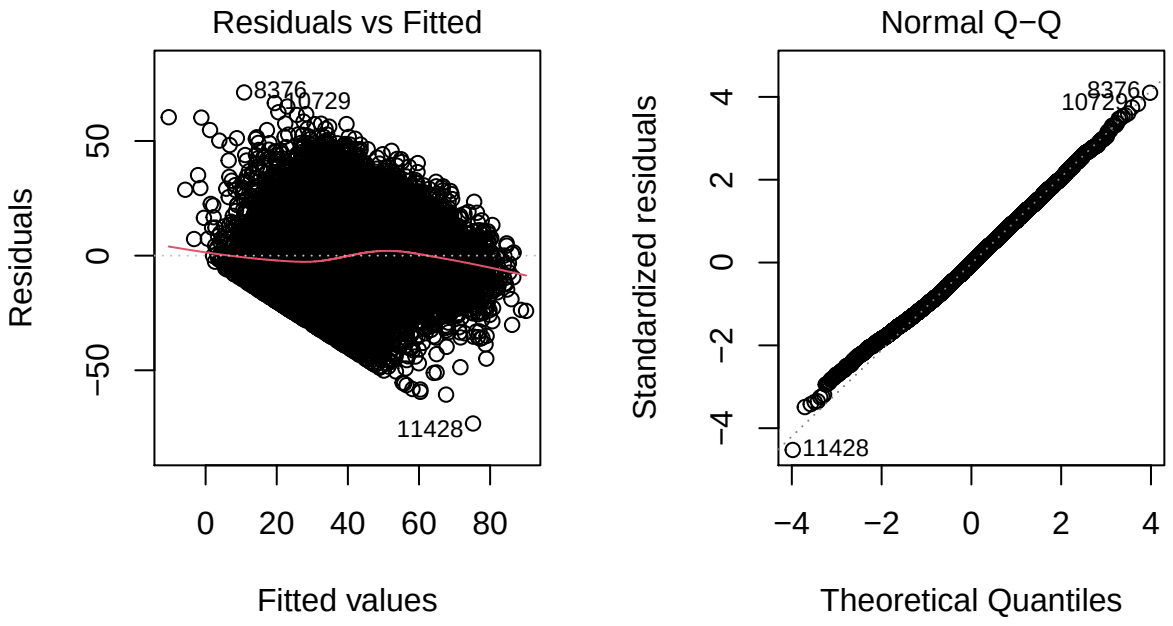
Results and Analysis

Training models

The pure linear model

From the exploratory data analysis, we take out the predictor variables with no correlation to the response variable, as well as the other ones explained before in the EDA.

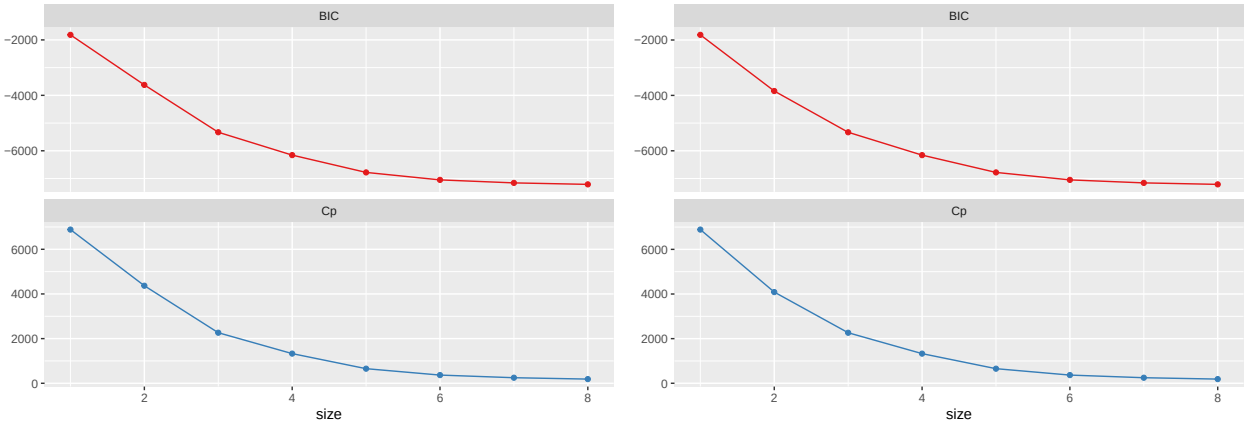
Then we train a linear regression model on the training set, and show its diagnostic plots:



The interesting part of the full model is that all predictor variables end up with a very low p-value in the summary of the full model, meaning they all are quite significant in some way when it comes to the response. This is important, as this builds the basis for predictor selection in other reduced models. From our Normal Q-Q plot, it would be reasonable to say the residuals resemble somewhat of a normal distribution, as many points lie near or on the linear line.

As we want to find some optimal combination of predictor variables, we try forward and backwards selection to get some examples. The graphs shown are the BIC and Cp for forward selection and backwards selection of predictors respectively:

```
## Warning: package 'leaps' was built under R version 4.0.3
```



From both stepwise selection methods, we can see that BIC and Cp both decrease when having more

predictors. This may mean that models with more predictors are much more favored in terms of BIC and Cp, gives us reason to look at modes with more predictor variables, and lines up with our observation of how good our predictors are (small p-value) in our full model.

We first try fitting a model recommended by the selection methods (check code)

For fun, we also fit a model based on choosing the variables with a p-value of less than 2e-16 from the full model (check code for predictors):

Something noteworthy upon closer inspection is that the reduced model recommended by our stepwise selection methods is very similar to our model based on choosing the predictors with the lowest magnitude of p-values.

Finally, we compare our trained models by using it to predict those in our validation set. We will use MSE as our CV score since we are focused on predictive power:

```
##            [,1]               [,2]                    [,3]
## mse_names  "full model mse"   "reduced model 1 mse" "reduced model 2 mse"
## model_mses "693.764862778526" "690.870681486223"    "689.512893800188"
```

From our choices of linear models, we can see that the reduced model of taking all the predictors that had less than 2e-16 for their p-value in the full model performed the best in terms of prediction on our validation set. However, the difference is very marginal. Our best model has an MSE of 689.5128.

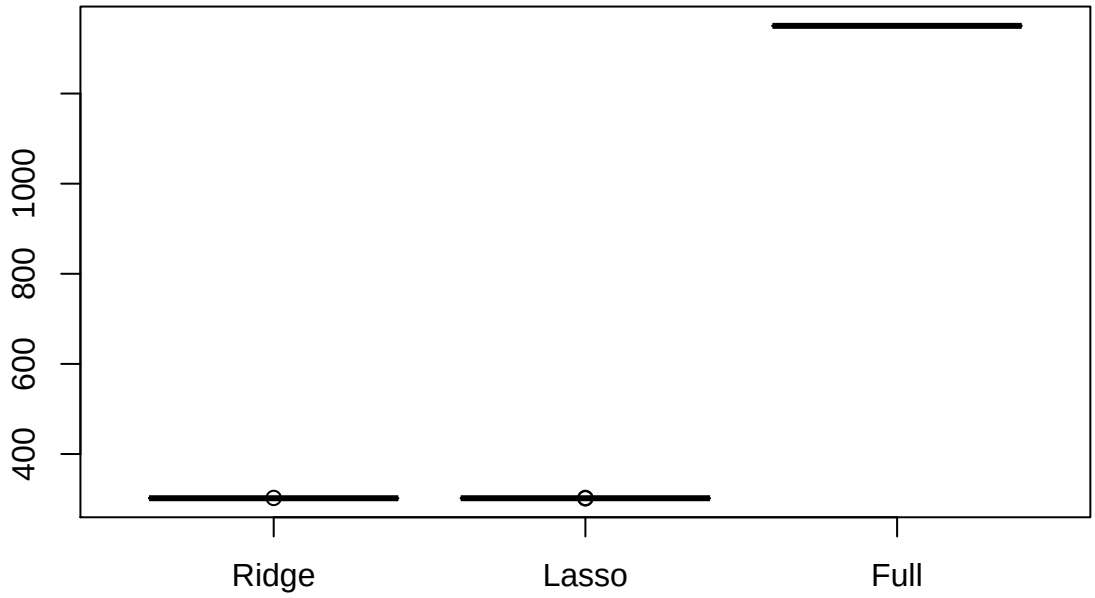
Ridge & LASSO Regression Model

FOr further comparison, we also ran variations of regression - LASSO and Ridge regression to find a model that might be better suited. So, as per the EDA, we divided our dataset.

```
## Warning: package 'glmnet' was built under R version 4.0.5

## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 4.0.5

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-2

## Warning: package 'MASS' was built under R version 4.0.5

##
## Attaching package: 'MASS'

## The following object is masked from 'package:patchwork':
##
##     area
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

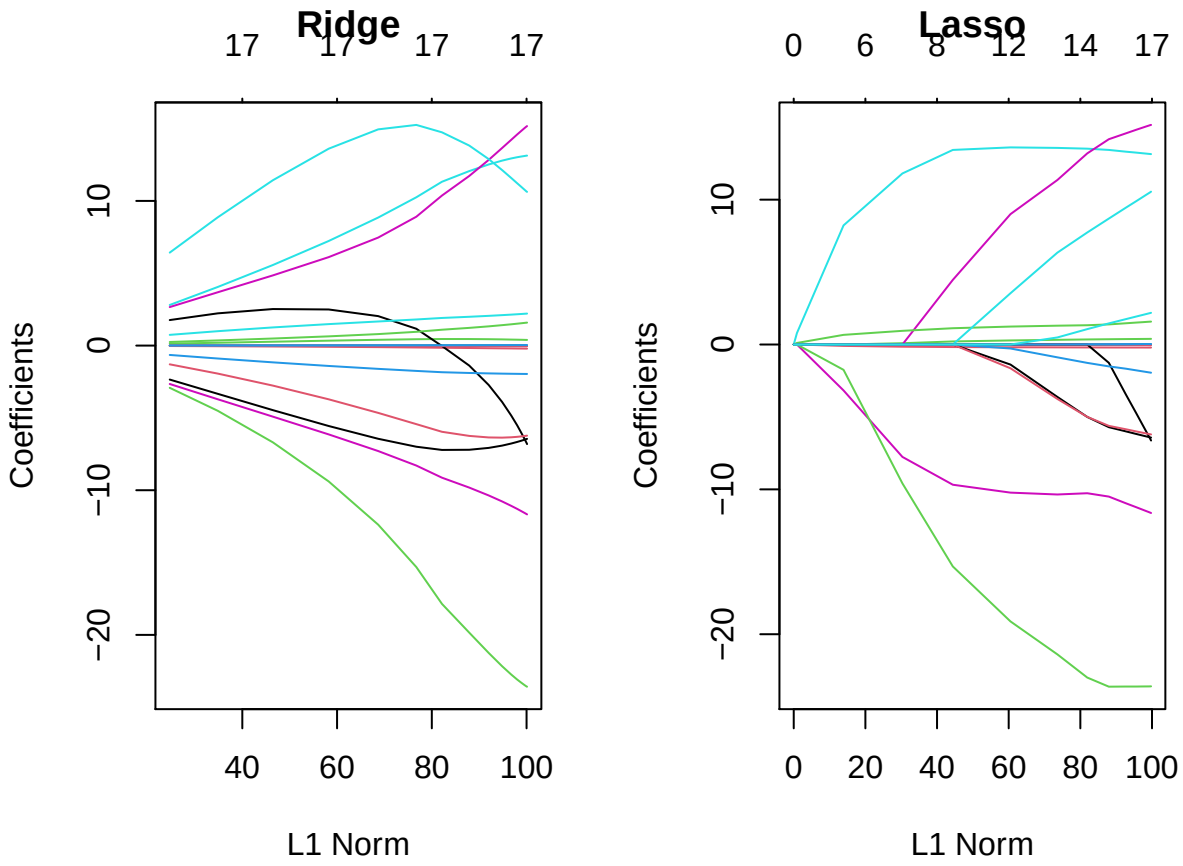
We ran 50 runs of 5-fold CV to tune the hyperparameters and achieve Ridge and LASSO models that would give us the best fit. The resulting boxplot compared the errors present within the the two models and the full model, to illustrate that the ridge model performed better than the other two, albeit only marginally outperforming LASSO.



#plot of regressions

```
par(mfrow = (c(1, 2)))
plot(reg_ridge$glmnet.fit, main = "Ridge")
# abline(v = sum(abs(coef(ridge.glmnet))))

plot(reg_lasso$glmnet.fit, main = "Lasso")
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```




```
# abline(v = sum(abs(coef(lasso.glmnet))))
```

```
# make predictions with regression models
validation_data <- read.csv("validation.csv")
vald_data_clean <- dplyr::select(validation_data, -X, -performer,
    -song, -spotify_genre, -spotify_track_album, -date)
x_vald <- as.matrix(vald_data_clean[, names(vald_data_clean) !=
    "spotify_track_popularity"])
y_vald <- as.vector(vald_data_clean$spotify_track_popularity)
```

We computed the predictive power of our models by comparing the models fitted to the validation set and capturing the MSE value on the new dataset.

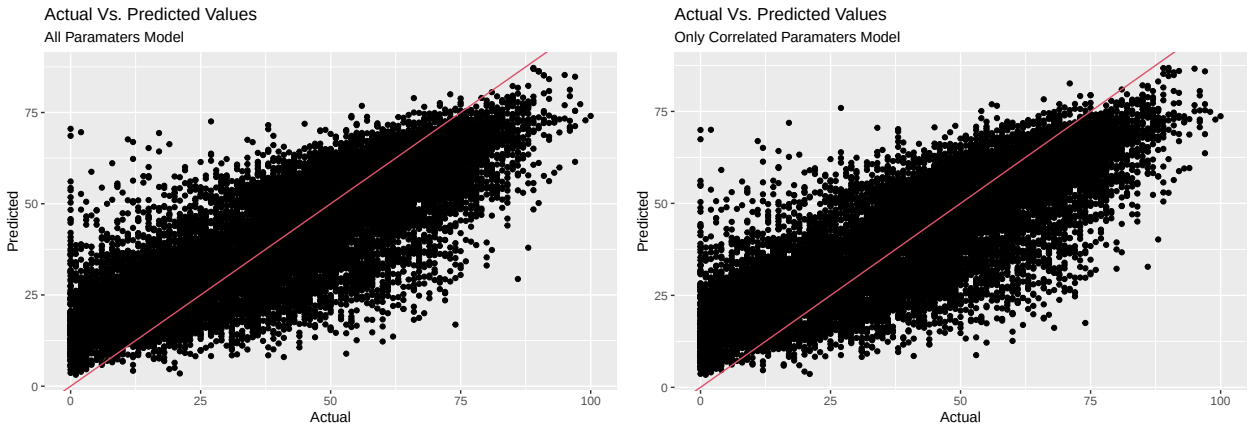
```
# evaluate model predicitons
ridge_pred <- predict(reg_ridge, newx = x_vald, s = "lambda.min")
lasso_pred <- predict(reg_lasso, newx = x_vald, s = "lambda.min")
ridge_mse <- mse(y_vald, ridge_pred)
lasso_mse <- mse(y_vald, lasso_pred)

test_data <- read.csv("test.csv")
test_data_clean <- dplyr::select(test_data, -X, -performer, -song,
    -spotify_genre, -spotify_track_album, -date)
```

```
x_test <- as.matrix(test_data_clean[, names(test_data_clean) !=
    "spotify_track_popularity"])

# evaluate model predicitons
ridge_pred.test <- predict(reg_ridge, newx = x_vald, s = "lambda.min")
lasso_pred.test <- predict(reg_lasso, newx = x_vald, s = "lambda.min")
```

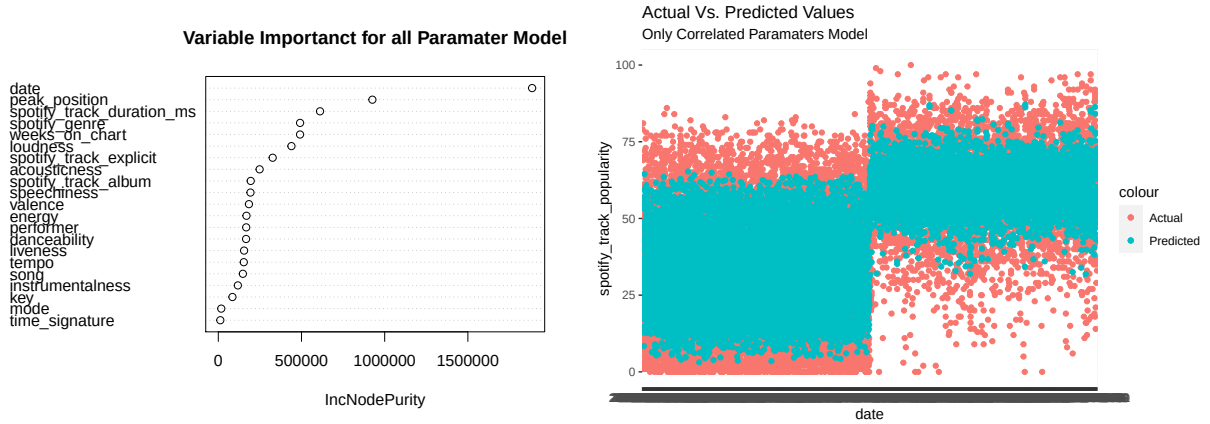
At the end we saw that the ridge regression performed better than the other models, however the regular linear regression seemed to have performed even better.

Random Forest Model

As we've seen in the EDA across the scatterplots, it is not immediately clear that a linear model fits our dataset very well. Since random forests are one of the best "out of the box" models, we will attempt to use random forests to predict the popularity of our songs. We will compare two different models, one with all fields and one with those with 0 correlation to our response removed.



As we can see in the above graphics, where we plotted the expected values versus the predicted values against one another, we get a roughly linear line along the y = x line shown in red. The models do not differ greatly between one another. We do however see that at the very high/low values our random forest predicts poorer than it does for values in the center. The mean squared error for the model with all parameters is 183.3488688 and for the model removing the uncorrelated parameters is 182.6269058.



We can see from the above plots that date is by far the biggest contributor to the prediction. This is followed by the peak position on the billboard chard, genre, duration, and weeks on the billboard charts.

These all make a lot of sense as the large contributors, the largest surprise to me was date beating billboard peak position, as it would make sense that billboard performance would indicate the Spotify popularity. In the next plot, date, which is clearly not a linear parameter with respect to the popularity is decently well predicted by our random forest. Though as expected from the previous plots, the data is more centered rather than covering the high/low values very well.

Comparing the models

The following table compares the mean squared error across the four models we've tested, with the results tested across a hold out validation set:

Random Forest	Linear Regression	LASSO	RIDGE
204.7491	689.5129	296.8817	296.8967

Here we see that random forest is a significantly better predictor than the other linear models.

Putting them all together

From the previous section, we see there is quite a difference between the predictions of all models. Here we will attempt to combine these models to see if they perform better together than separately. We will try two different methods: a simple average across the 4 models, as well as using linear regression to assign weights among the models.

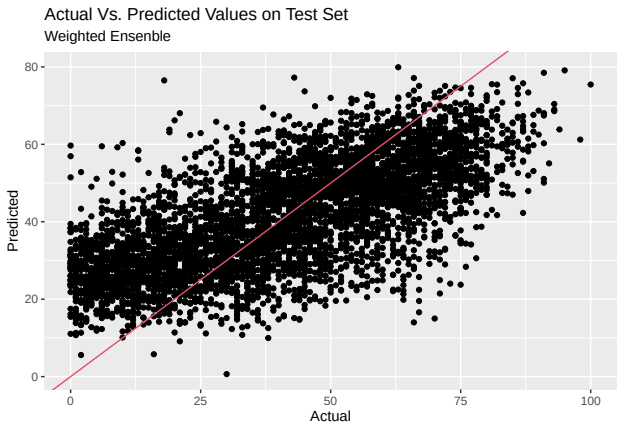
Our weighted average produced weights:

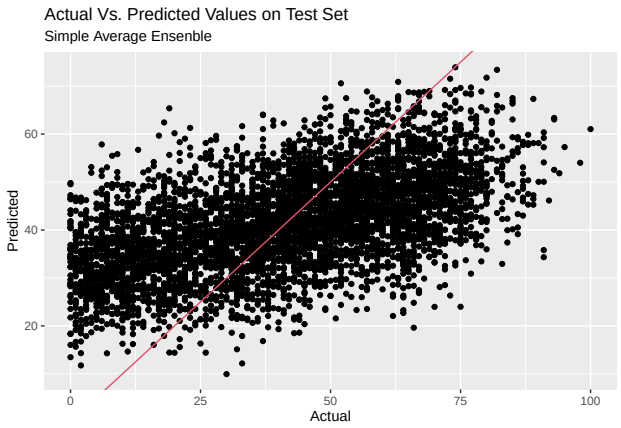
```
##         (Intercept)     `Random Forest` `Linear Regression`               LASSO
##          -3.5843999           0.9744643          -0.3003722          17.2830934
##               RIDGE
##         -16.8944742
```

Using these different models, we see that the weighted average produces a mean squared error on our validation set of 201.91, and the simple average produces a MSE of 255.01.

Results and Discussion

Results





Random Forest	Linear Regression	LASSO	Ridge	Mean Ensembling	Weighted Ensembling
199.9431	288.9258	694.7442	694.4446	357.0706	267.5161

Discussion

Results Discussion

Since all the models were regressive models, there isn't an accuracy parameter to compare. However, we can compare the MSE values to check for the predictive power of the models. Between the multiple models, the best model was the Random Forest with an MSE of 199.117. Surprisingly, the random forest outperformed the ensemble methods and this could largely due to the comparatively terrible LASSO and Ridge MSE values. Even after the weightage of the different models, ensembling came in second with an MSE of 263.23.

Limitations and Further Work

We can see that the general accuracy of results increase as the complexity of our models increased. This tells us that the true model may follow a non-linear trend. The predictive capabilites could . So, it is worth exploring using other non-linear models such as polynomial regressions or neural networks.