Our Music Playlists

Search

Home

Library

MGSC 310

Stream Prediction Analysis

Aylo Corshen, Karla Carmona, Alex Rea, Samantha Waters

Content

Problem	What is our business use case. Where do we find a need for data analytics in Spotify?
Data Set	What data we used. How we cleaned it.
Exploratory Graphs	Initial insights
Logistic and Lasso Regression	Trying out our data set with these two methods. Comparing if they worked.
Decision Tree and Random Forest	Predicting counted stream
Conclusions	Our results and recommendations

Business Objective

- Our models are designed to help managers and artists determine song characteristics that lead to a listening length of over 30 seconds (a counted stream)
- Benefits:
  - Financial upside
  - Increase success in the eyes of Labels

The Data Set:

Link Here

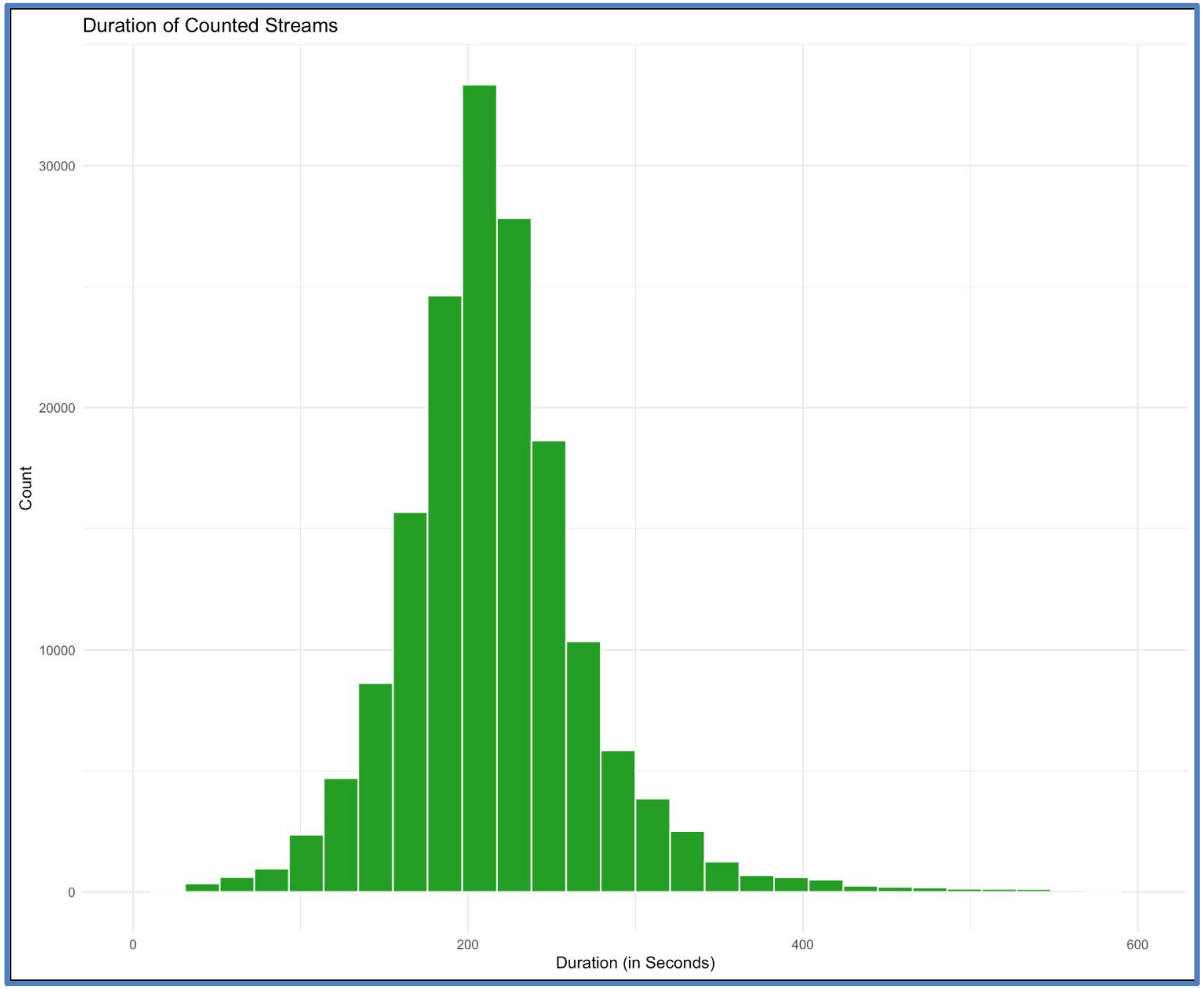
- counted_stream
- us_popularity_estimate
- acousticness
- beat_stregnth
- danceability
- dyn_range_mean
- energy
- flatness
- instrumentalness
- valence
- key
- liveness
- loudness
- mechanism
- mode
- organism
- speechiness
- tempo
- time_signature
- Duration

LET’S TALK DATA

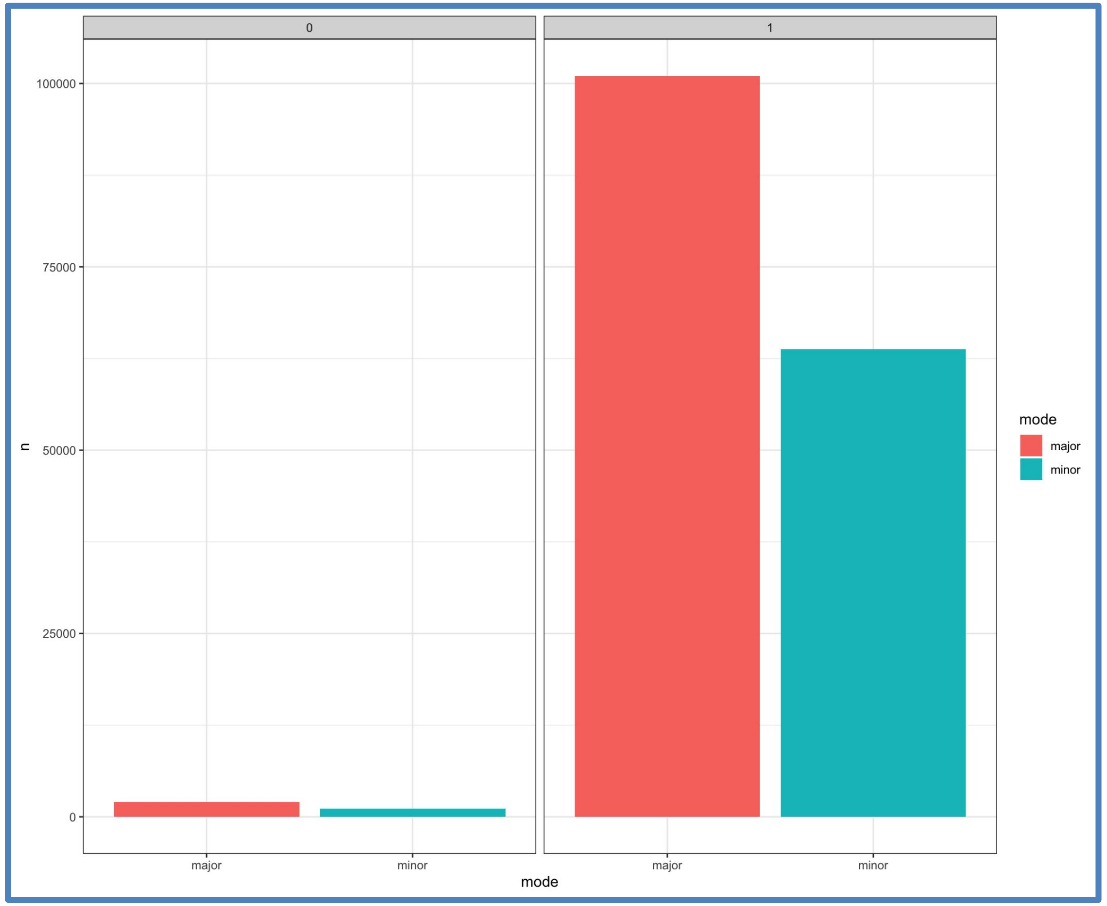
Summary Statistics

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
duration	167880	215.893	60.224	30.013	183.503	240.8	1787.761
us_popularity_estimate	167880	99.745	0.893	90.019	99.914	99.999	100
acousticness	167880	0.218	0.246	0	0.031	0.337	0.996
beat_strength	167880	0.548	0.159	0	0.434	0.666	0.99
bounciness	167880	0.58	0.179	0	0.452	0.724	0.973
danceability	167880	0.669	0.159	0	0.562	0.787	0.985
dyn_range_mean	167880	9.098	2.54	0	7.225	10.765	32.343
energy	167880	0.627	0.182	0	0.514	0.761	1
flatness	167880	1.008	0.039	0	0.989	1.034	1.103
instrumentalness	167880	0.032	0.143	0	0	0	0.999
key	167880	5.203	3.682	0	1	8	11
liveness	167880	0.19	0.151	0	0.1	0.236	0.996
loudness	167880	-7.05	3.164	-60	-8.232	-5.083	1.634
mechanism	167880	0.597	0.208	0	0.453	0.758	1
mode	167880						
... major	103063	61.4%					
... minor	64817	38.6%					
organism	167880	0.348	0.189	0	0.205	0.47	0.962
speechiness	167880	0.142	0.133	0	0.045	0.206	0.961
tempo	167880	122.675	29.822	0	97.005	144.073	218.775
time_signature	167880						
... 0	39	0%					
... 1	1017	0.6%					
... 3	7546	4.5%					
... 4	157101	93.6%					
... 5	2177	1.3%					
valence	167880	0.459	0.229	0	0.28	0.626	1
counted_stream	167880						
... 0	3146	1.9%					
... 1	164734	98.1%					

Exploratory Data Analysis



Major Vs. Minor in Counted/Not-Counted Streams

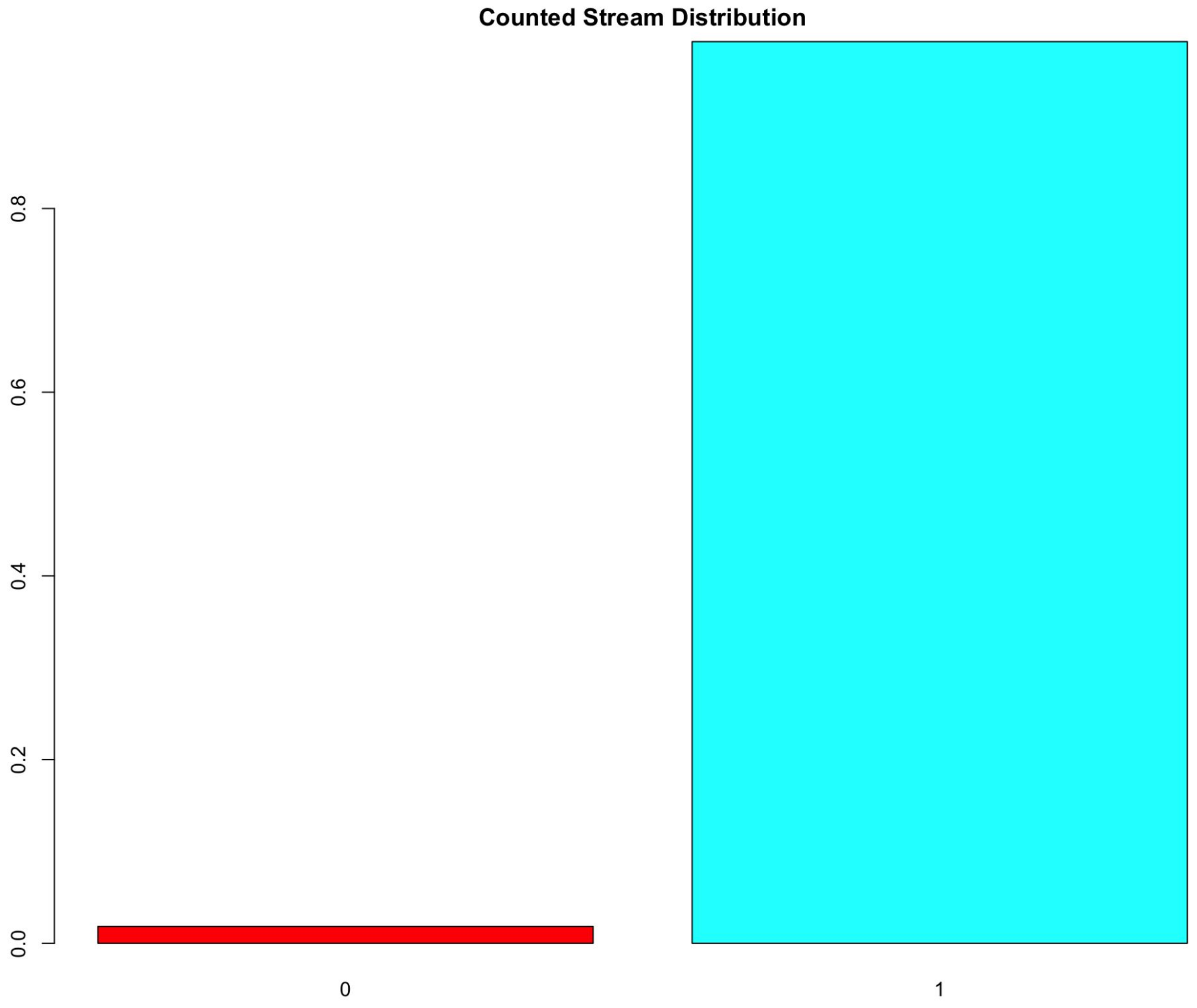


Data Cleaning

```
#Merging song and user data sets together
songs <- merge(track_features, session_logs, by="track_id")

#omitting any NA value rows just in case
songs_clean <- na.omit(songs)
colSums(is.na(songs_clean))

#turning variable into binary session
songs_clean <- songs_clean %>%
  mutate(skip_1 = ifelse(skip_1 == "TRUE", 1,0),
          skip_2 = ifelse(skip_2 == "TRUE", 1,0),
          skip_3 = ifelse(skip_3 == "TRUE", 1,0),
          not_skipped = ifelse(not_skipped == "TRUE", 1,0))

#creating stream counted variable
songs_clean$stream_counted <- ifelse((songs_clean$skip_3 == 1 |
songs_clean$not_skipped == 1), 1, 0)
```

- Merging datasets
  - song dataset - song characteristics
  - user dataset - user session characteristics
- removing NA values
- converting to binary
- creating variable determining whether the song was skipped or no (“stream_counted”)

Data Cleaning Cont.



- Class imbalance
- Extremely overtrained
- Deceitful models
- Resampling solution

Undersampling

```
table(songs_clean$counted_stream)
```

```
[1] "factor"

   0      1
3146 164734
```

- There are 3146 total observations
- Created a sample from our data that undersamples our total data frame
- New total of 6292 observations

```
under <- ovun.sample(counted_stream ~ . , data = songs_clean, method = "under", N = 6292)$data
table(under$counted_stream)
```

```
   1    0
3146 3146
```

Logistic Regression

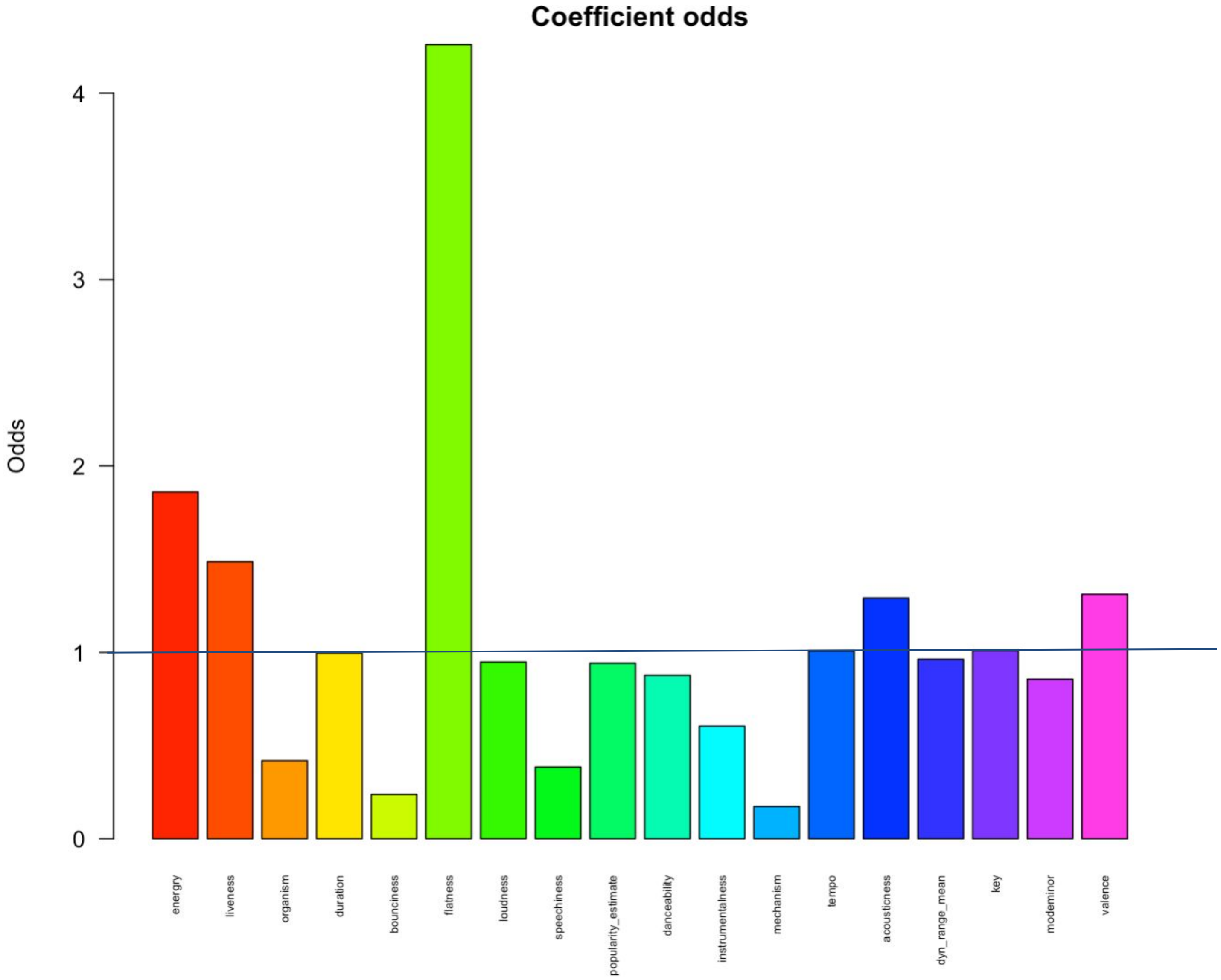
	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-7.0020038	197.0052229	-0.036	0.971647	
duration	-0.0035284	0.0005573	-6.331	0.000000000244	***
us_popularity_estimate	-0.0738231	0.0334985	-2.204	0.027540	*
acousticness	0.4264435	0.4943988	0.863	0.388385	
beat_strength	4.4317751	1.1527953	3.844	0.000121	***
bounciness	-2.3352897	1.4572264	-1.603	0.109032	
danceability	0.2978882	0.5211672	0.572	0.567607	
dyn_range_mean	-0.0621063	0.0519757	-1.195	0.232122	
energy	0.6118638	0.3601259	1.699	0.089314	.
flatness	2.6746740	1.4345547	1.864	0.062257	.
instrumentalness	-0.4318761	0.2433486	-1.775	0.075944	.
key	0.0000261	0.0083984	0.003	0.997521	
liveness	0.3986247	0.2112599	1.887	0.059175	.
loudness	-0.0488163	0.0151427	-3.224	0.001265	**
mechanism	-1.9637104	0.7548990	-2.601	0.009287	**
modeminor	-0.0769325	0.0641281	-1.200	0.230268	
organism	-1.5565586	1.1693703	-1.331	0.183153	
speechiness	-0.6628714	0.3266690	-2.029	0.042440	*
tempo	0.0058844	0.0014245	4.131	0.000036116319	***
time_signature1	12.3710457	196.9729597	0.063	0.949921	
time_signature3	11.6267741	196.9724747	0.059	0.952930	
time_signature4	11.7430807	196.9724728	0.060	0.952460	
time_signature5	11.5683536	196.9727077	0.059	0.953167	
valence	0.3933686	0.1634770	2.406	0.016117	*

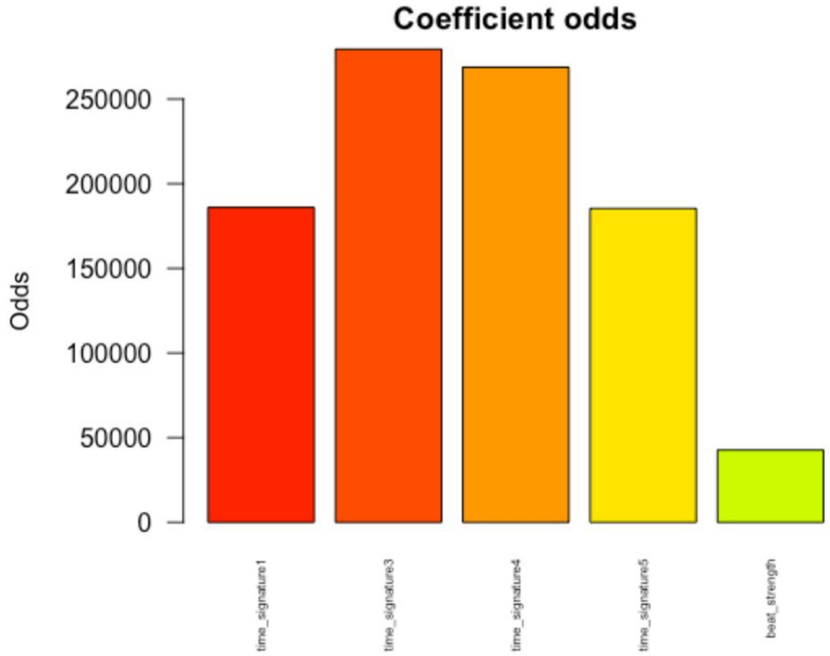
Exponentiating Odds

(Intercept)	duration	us_popularity_estimate	acousticness
0.0009100566	0.9964778473	0.9288359817	1.5317999997
beat_strength	bounciness	danceability	dyn_range_mean
84.0805392288	0.0967824379	1.3470112188	0.9397830241
energy	flatness	instrumentalness	key
1.8438647789	14.5076200290	0.6492898153	1.0000260957
liveness	loudness	mechanism	modeminor
1.4897744657	0.9523560771	0.1403367525	0.9259523552
organism	speechiness	tempo	time_signature1
0.2108604792	0.5153693682	1.0059017759	235872.2588995966
time_signature3	time_signature4	time_signature5	valence
112058.2505202505	125879.5379400867	105699.3099984378	1.4819645055

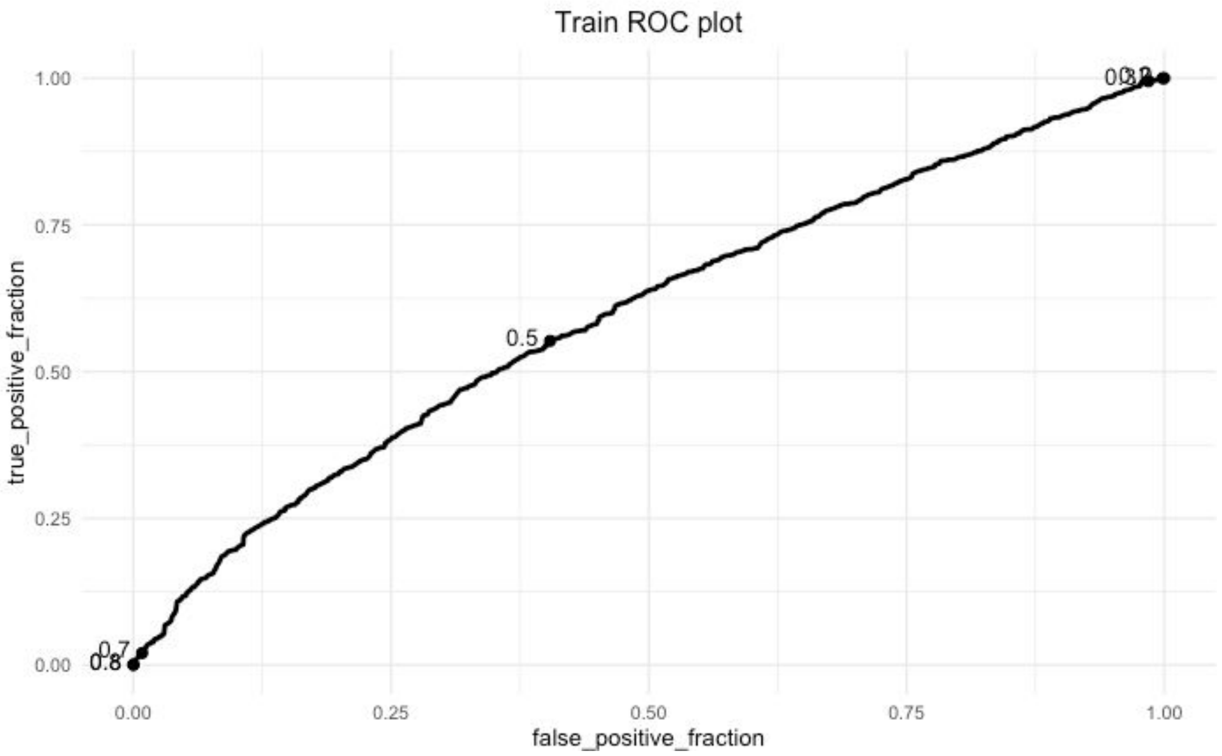
- A 1 unit increase (1 second) in duration will lead to a .4% decrease in the likelihood of a stream counting
- A 1 unit increase in tempo (1 bpm) will increase the likelihood of a stream being counted by .5%

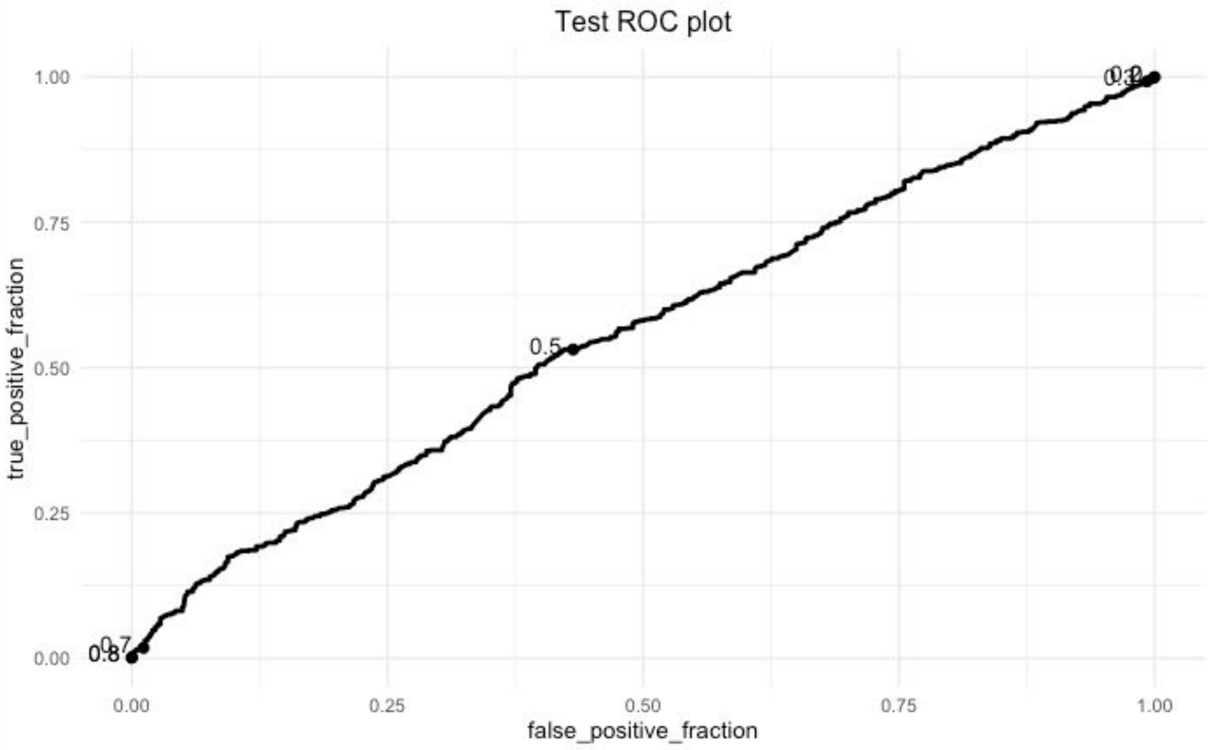
Visualizing Odds





ROC plot training and testing



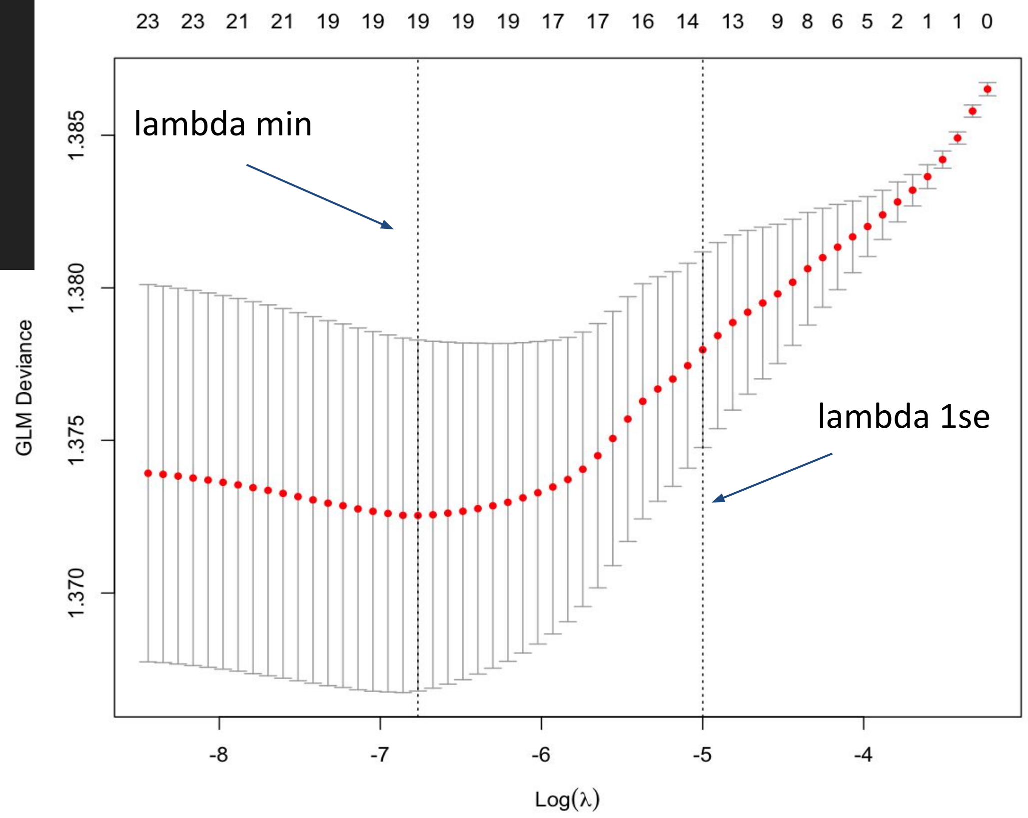


AUC <dbl>
0.5975678

AUC <dbl>
0.5581493

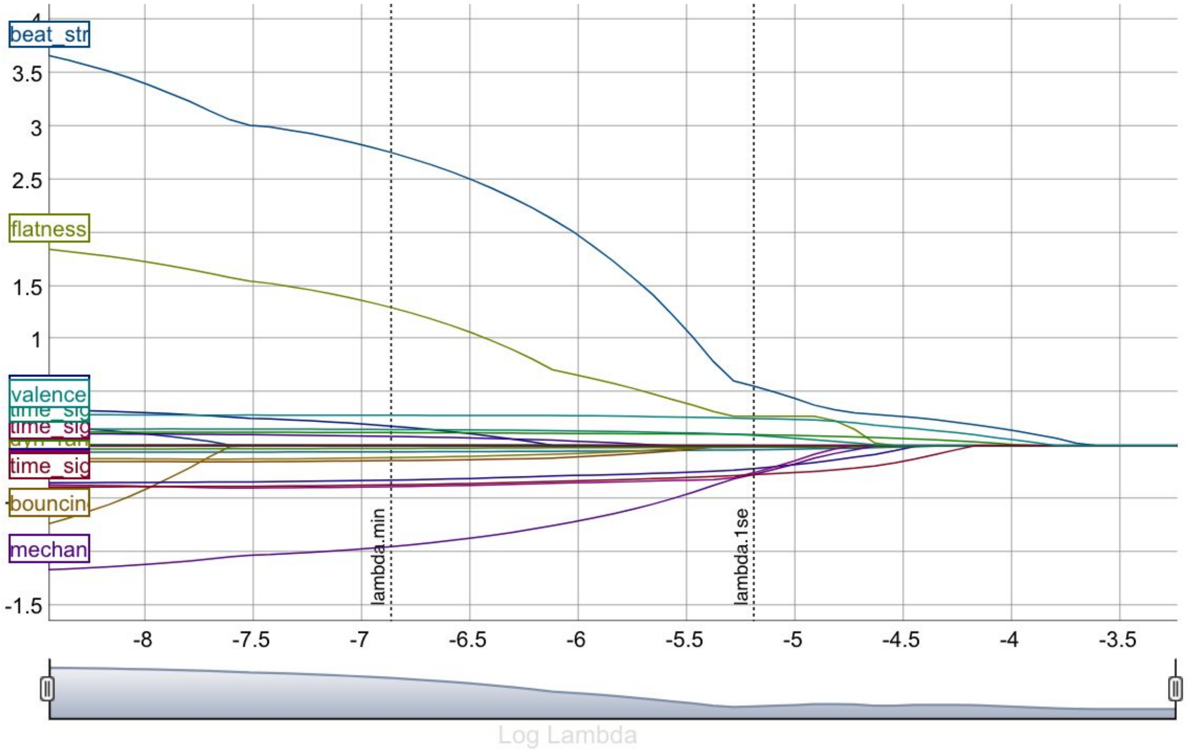
LASSO

```
#Lasso Model
lasso_mod <- cv.glmnet(counted_stream ~ .,
                        data = songs_train,
                     family = binomial,
                       # note alpha = 1 sets lasso, setting to 1!
                       alpha = 1)

# Final model with lambda.1se
lasso_1se <- glmnet(counted_stream ~ .,
                   data = songs_train,
                   family = binomial,
                   alpha = 1,
                   lambda = lasso_mod$lambda.1se)
```

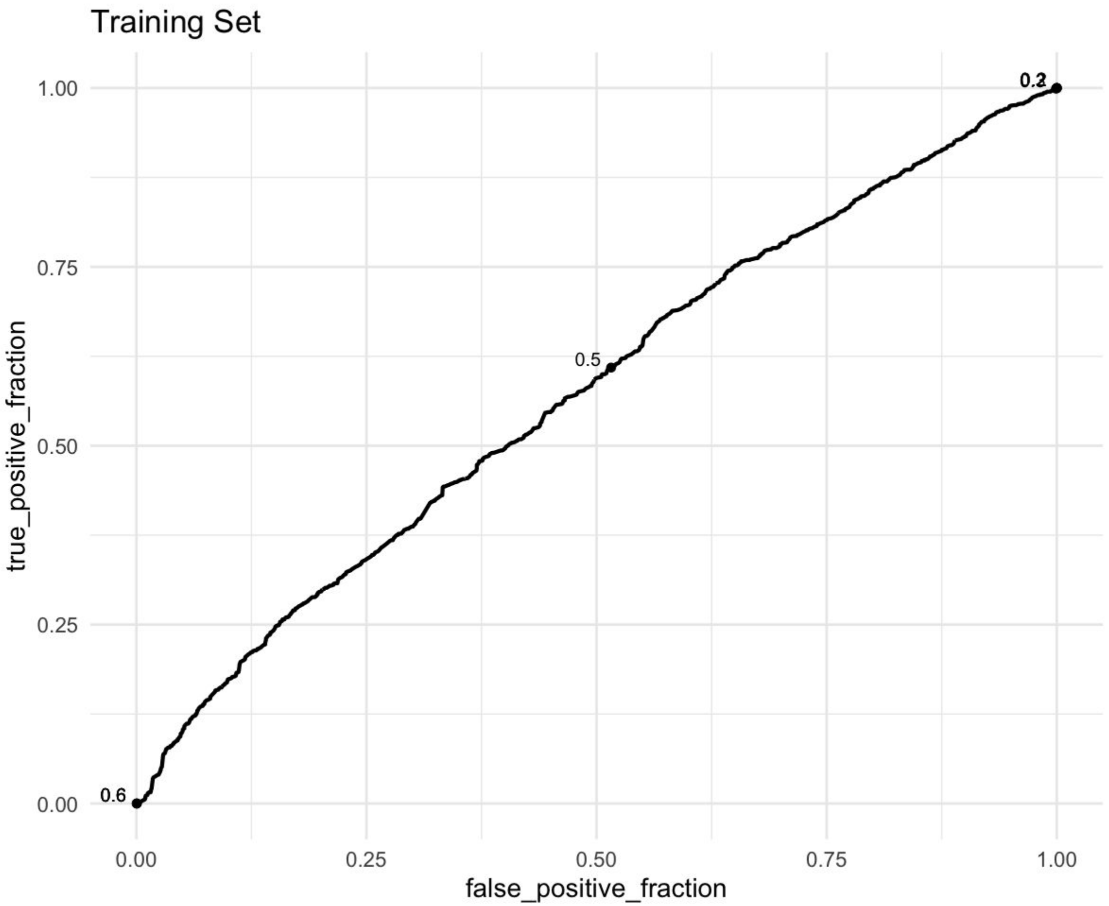


LASSO

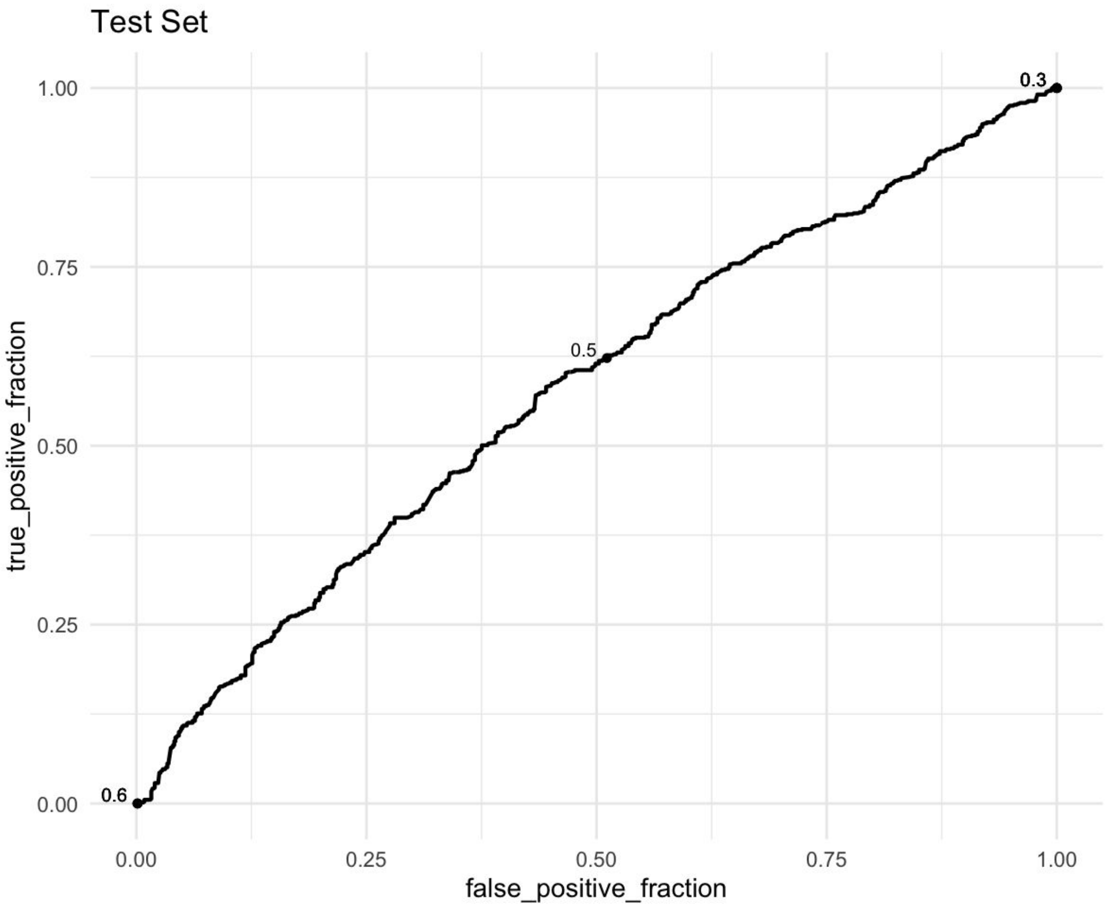
	s1	Exp. s1
(Intercept)	3.034	20.7801873
duration	-0.002	0.9980020
us_popularity_estimate	-0.034	0.9665715
acousticness	.	1.0000000
beat_strength	0.445	1.5604902
bounciness	.	1.0000000
danceability	.	1.0000000
dyn_range_mean	.	1.0000000
energy	.	1.0000000
flatness	0.275	1.3165307
instrumentalness	-0.180	0.8352702
key	.	1.0000000
liveness	.	1.0000000
loudness	-0.015	0.9851119
mechanism	-0.175	0.8394570
modemajor	0.095	1.0996589
modeminor	0.000	1.0000000
organism	.	1.0000000
speechiness	-0.145	0.8650223
tempo	0.000	1.0000000
time_signature0	.	1.0000000
time_signature1	.	1.0000000
time_signature3	.	1.0000000
time_signature4	0.070	1.0725082
time_signature5	-0.254	0.7756918
valence	0.245	1.2776213



LASSO - ROC Plot

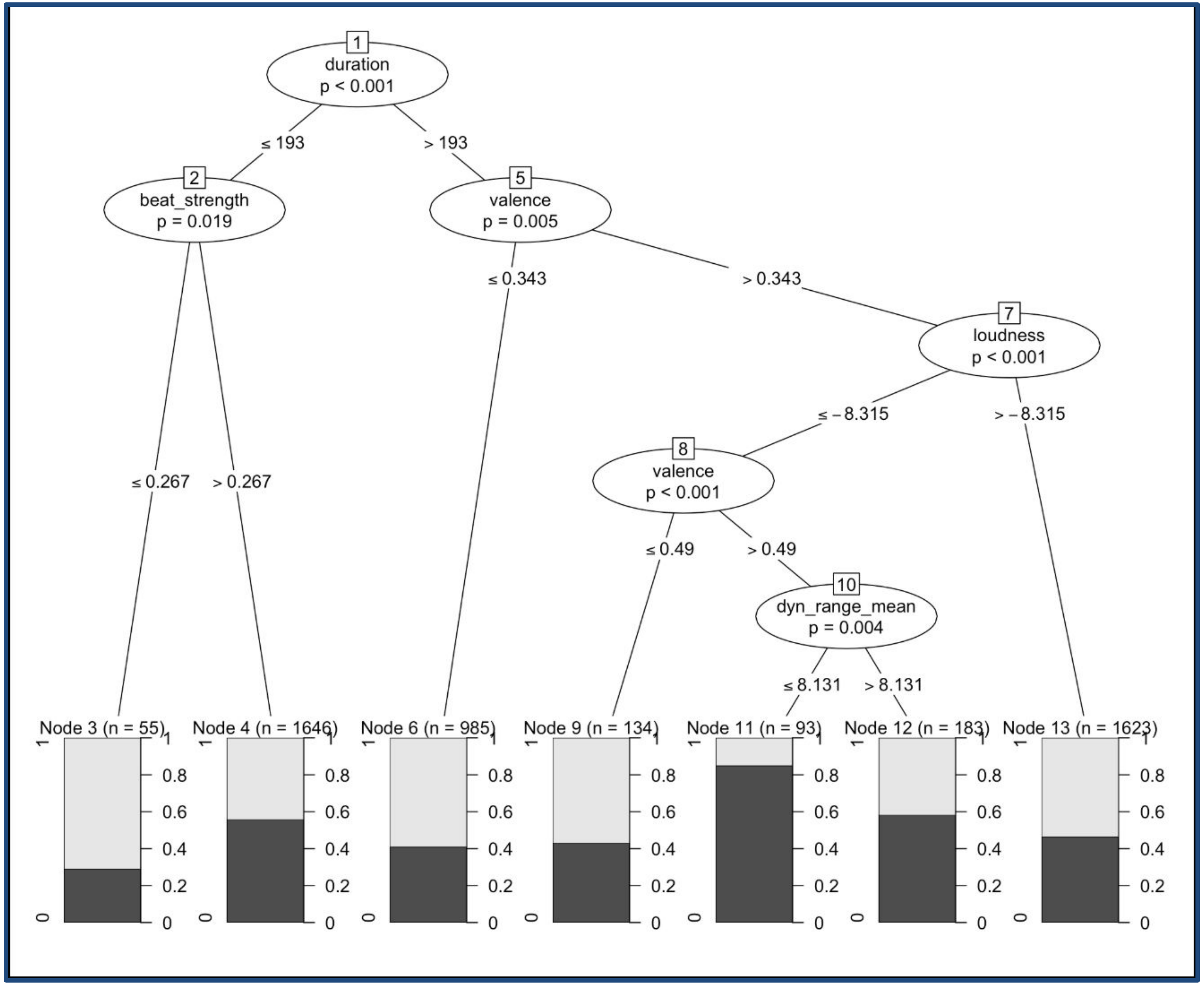


AUC <dbl>	AUC <dbl>
0.5745181	0.5975678
LASSO	Logistic

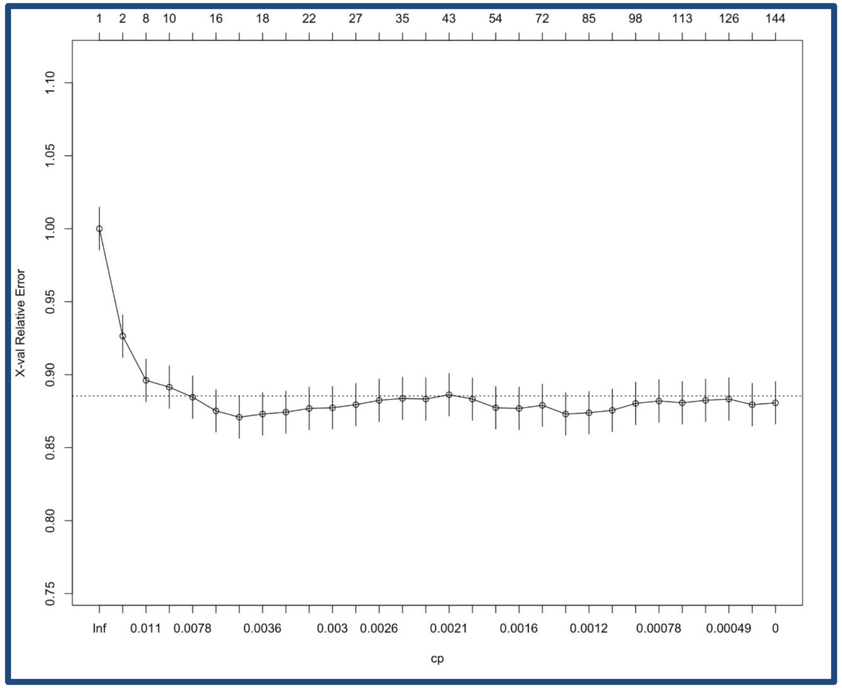


AUC <dbl>	AUC <dbl>
0.5792151	0.5581493
LASSO	Logistic

Decision Tree



Decision Tree Model Accuracy




```
pruned <- prune(mod_rpart, cp = 0.0038477982)
```

Pruned Tree

```
Confusion Matrix and Statistics

          Reference
Prediction   1   0
         1 540 467
         0 226 340

               Accuracy : 0.5594
                 95% CI : (0.5345, 0.5842)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : 0.000125
```

Original Tree

```
Confusion Matrix and Statistics

          Reference
Prediction   1   0
         1 528 456
         0 238 351

               Accuracy : 0.5588
                 95% CI : (0.5339, 0.5835)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : 0.0001521
```

Test

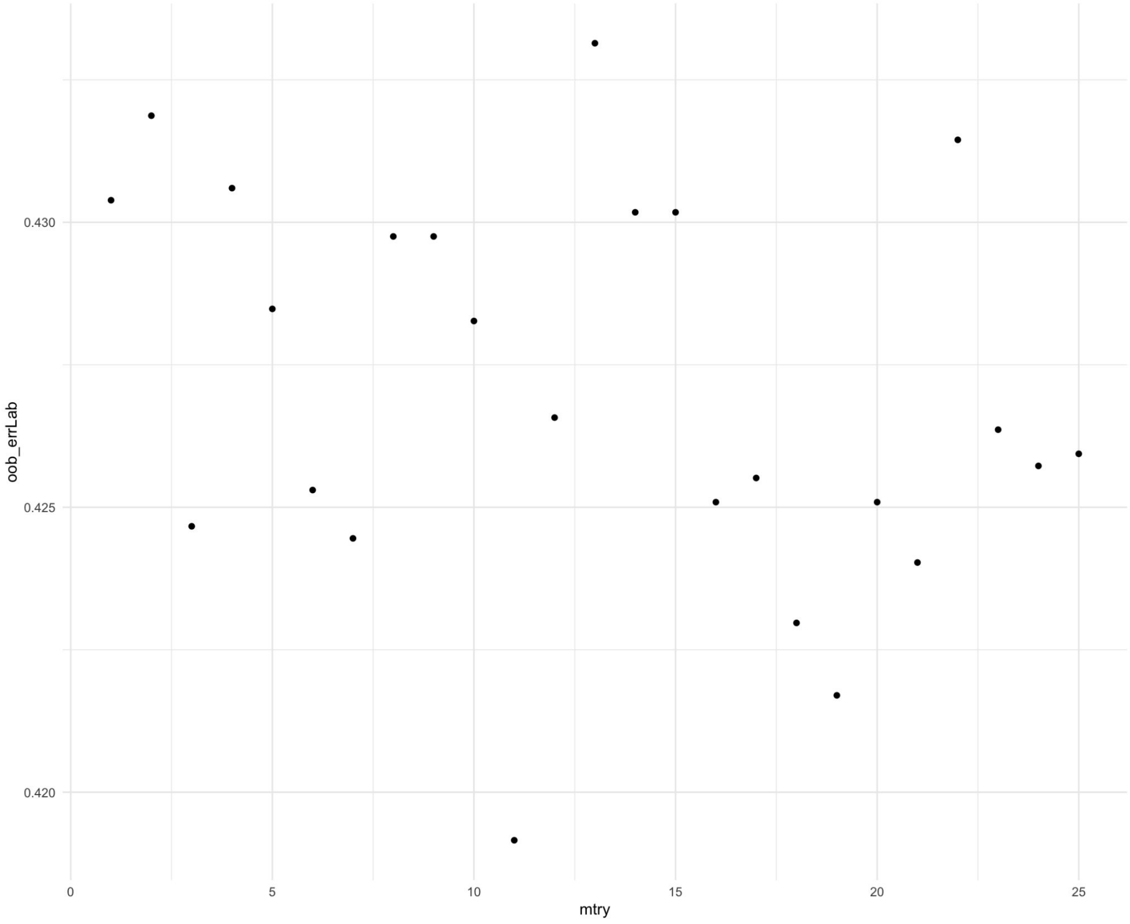
```
Confusion Matrix and Statistics

          Reference
Prediction    1    0
         1 1566 1231
         0  814 1108

               Accuracy : 0.5666
                 95% CI : (0.5524, 0.5808)
    No Information Rate : 0.5043
    P-Value [Acc > NIR] : < 0.00000000000000022
```

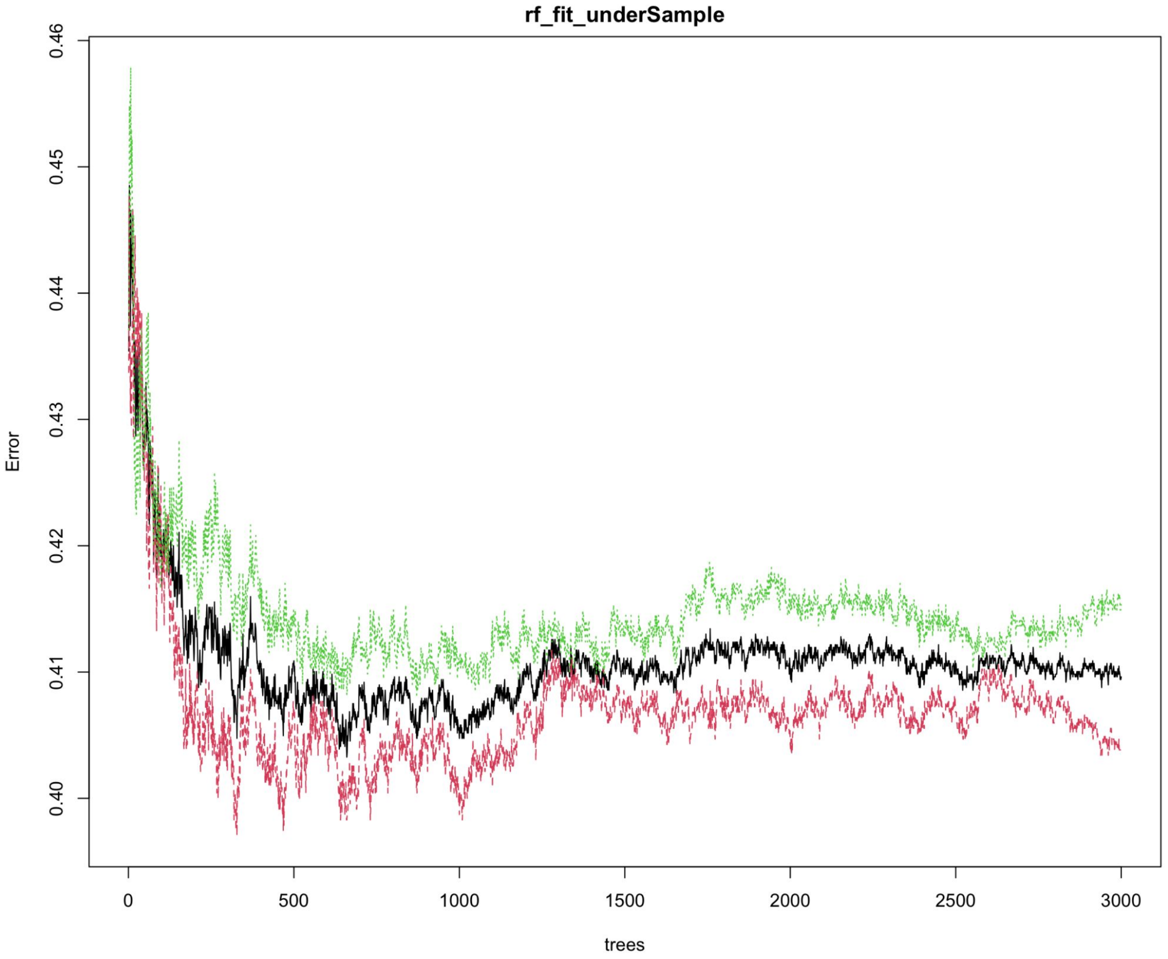
Train

Using a Random Forest (Thought Process)

- Hyperparameter Tuning
  - Appropriate number of variables to sample from
  - Appropriate number of trees
- Random Forest using complete data vs. Random Forest using sample
- Accuracy Comparison
- Interpretability

Hyperparameter Tuning





- Finding the correct number of variables to sample from
- What amount of trees should we use?

Using a Random Forest to Predict a Stream (OOB)

```
Call:
 randomForest(formula = counted_stream ~ ., data = songs_train,      mtry = 11,
importance = TRUE, na.action = na.roughfix)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 11

        OOB estimate of  error rate: 2.07%
Confusion matrix:
    0      1 class.error
0  88   2220 0.961871750
1 386 123216 0.003122927
```

- Random Forest without undersampling

```
Error: vector memory exhausted (limit reached?)
```

```
Call:
 randomForest(formula = counted_stream ~ ., data = songs_train_under,      mtry = 11,
ntree = 1500, importance = TRUE, na.action = na.roughfix)
               Type of random forest: classification
                     Number of trees: 1500
No. of variables tried at each split: 11

        OOB estimate of  error rate: 42.28%
Confusion matrix:
     1    0 class.error
1 1561  807   0.3407939
0 1188 1163   0.5053169
```

- Random Forest with undersampling

Random Forest Confusion Matrix

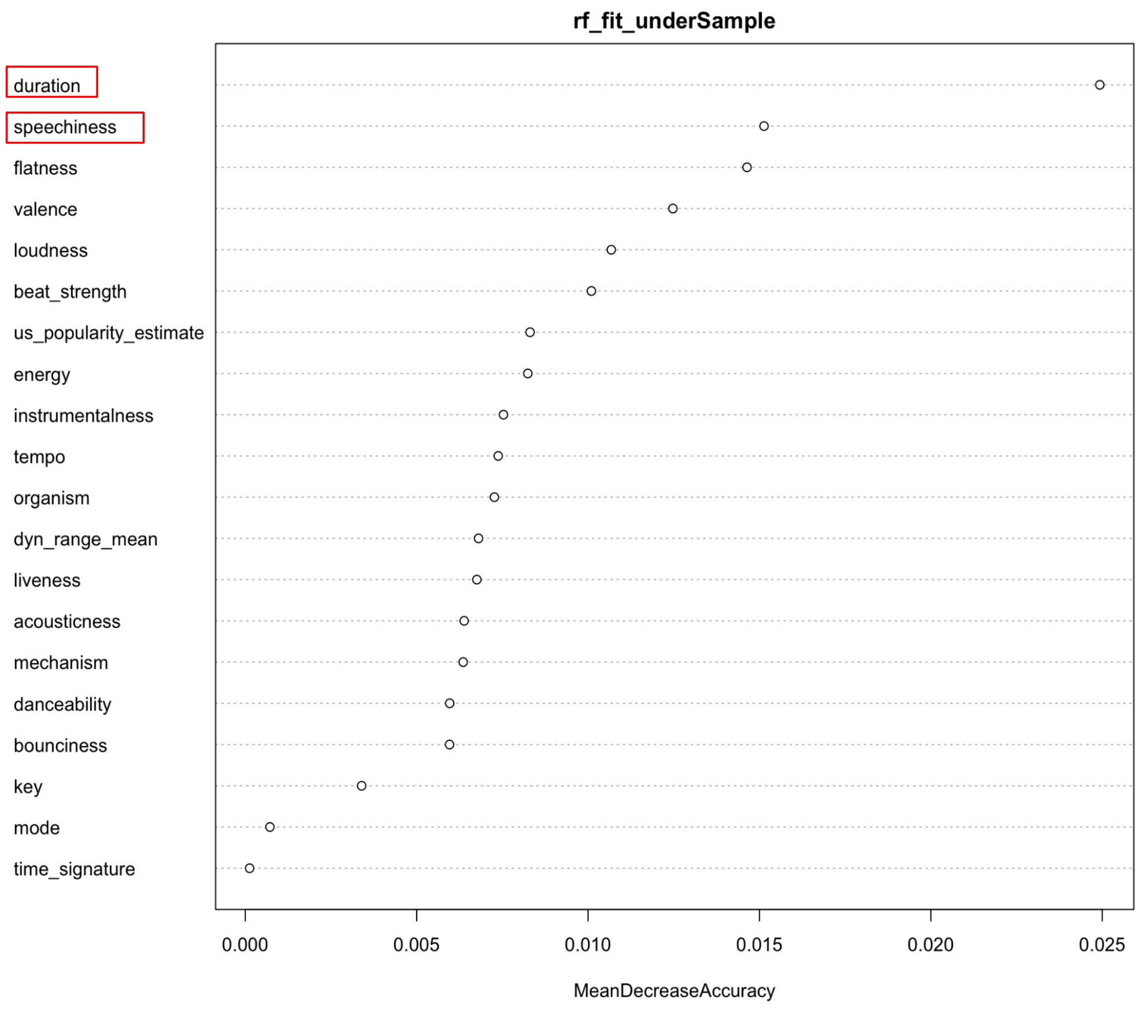
```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0    24   104
         1   814 41028

               Accuracy : 0.9781
                 95% CI : (0.9767, 0.9795)
    No Information Rate : 0.98
    P-Value [Acc > NIR] : 0.9972

                  Kappa : 0.0446

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.99747
            Specificity : 0.02864
         Pos Pred Value : 0.98055
         Neg Pred Value : 0.18750
             Prevalence : 0.98003
         Detection Rate : 0.97756
   Detection Prevalence : 0.99695
      Balanced Accuracy : 0.51306

       'Positive' Class : 1
```

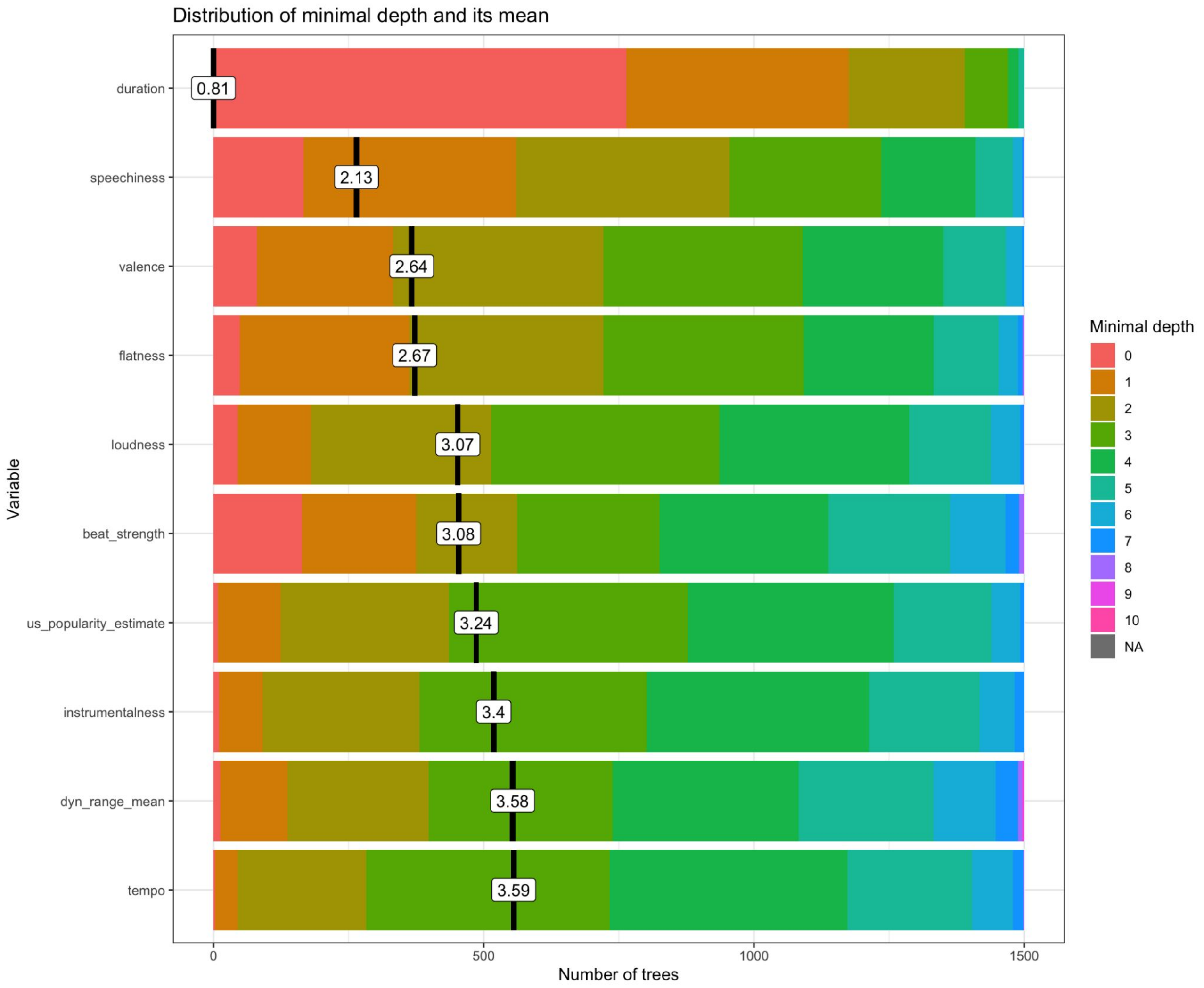
- Random Forest without undersampling

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0   595 13518
         1   159 27698

               Accuracy : 0.6741
                 95% CI : (0.6696, 0.6786)
    No Information Rate : 0.982
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0476

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.67202
            Specificity : 0.78912
         Pos Pred Value : 0.99429
         Neg Pred Value : 0.04216
             Prevalence : 0.98203
         Detection Rate : 0.65995
   Detection Prevalence : 0.66374
      Balanced Accuracy : 0.73057

       'Positive' Class : 1
```

- Random Forest using undersampling

Random Forest Interpretability





Would we recommend this model?

- While we saw a 67.41% accuracy rate with our random forest, we wouldn't recommend it for business application yet.
- The success of a track most likely has many more variables than we have available
  - ex: Different genres have different expectations on tempo, danceability, ect
  - making a one size fits all model for music is not advisable
- If we where to continue we would love to get a hold of data with more variables such as genre to make models that are more specific to an artist style.

Search

Home

Library

Thank you!

https://github.com/a-rea/mgsc310FinalProject