Predicting a Hit Song Using Machine Learning

Justin Ly 4269684

06/04/20

Introduction

This research project is a utilization of machine learning to delve into the constructs of popular music. Through recent history, a common criticism of mainstream music has been that all pop music sounds the same. Obviously they don't sound exactly the same to the average ear, but we may be able identify common trends among popular songs in areas such as acousticness, speed, or tonality of the songs. In response to this, our purpose in this project was to determine if we could develop a classifier that could accurately predict whether or not a song will have mainstream success based on various audio features.

All analyses performed in this project were done using the R software for statistical computing and graphics. In this supervised machine learning project, we implemented three different methods in developing our classfiier. We used logistic regression and classification trees to develop a basic classifier, and used an ensemble method by creating a random forest decision trees algorithm. To form our conclusions, the results of these classifiers were compared using test errors to assess accuracy and ROC curves to assess performance in order to choose the best model for our classifier.

Data Overview:

The dataset used is titled 'The Spotify Hit Predictor Dataset (1960-2019)' put together by Farooq Ansari and is taken from the website Kaggle: https://www.kaggle.com/theoverman/the-spotify-hit-predictor-dataset#dataset-of-10s.csv. The collection of data has 6 datasets, one for each decade from the 1960s to the 2010s, and each dataset has over 6000 song entries, with 19 features. The data was taken and compiled from Spotify's "Audio Features" website, where you can download audio features for any song on the platform. For this project we perform our analysis on the dataset from the 2010s, which contains 6398 songs.

The response variable from the data was created by the author is the binary variable 'target', which is a 0 if the song was not a hit, and a 1 if the song was a hit. The '0' designation is given to the song if the track or track artist didn't appear in the 'hit' list of the decade given by Billboard. A song is also given a 0 if it belongs to an avant-garde/non-mainstream genre. Otherwise, the song is designated a '1.'

The predictor variables used for model building are: danceability, energy, loudness, speechiness, acousticness, valence, tempo, and duration. All of these variables used are numeric based on predefined scales.

From the Kaggle data description:

danceability: Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

energy: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.

loudness: The overall loudness of a track in decibels (dB).

speechiness: Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.

acousticness: A confidence measure from 0.0 to 1.0 of whether the track is acoustic

tempo: The overall estimated tempo of a track in beats per minute (BPM)

duration_ms: The duration of the track in milliseconds

valence: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track

Methods:

Data Processing

The data set comes with 19 features, many of which we deem unneccessary and irrelevant. We cut down the list to 8 features after removing variables such as song title, artist, time signature, and key. Although there would be common trends in some of the variables cut, the variance of the data in these features wouldn't provide interesting results.

Furthermore we found using is.na() that there are no missing values in the data, so there were no issues in the completeness of the dataset.

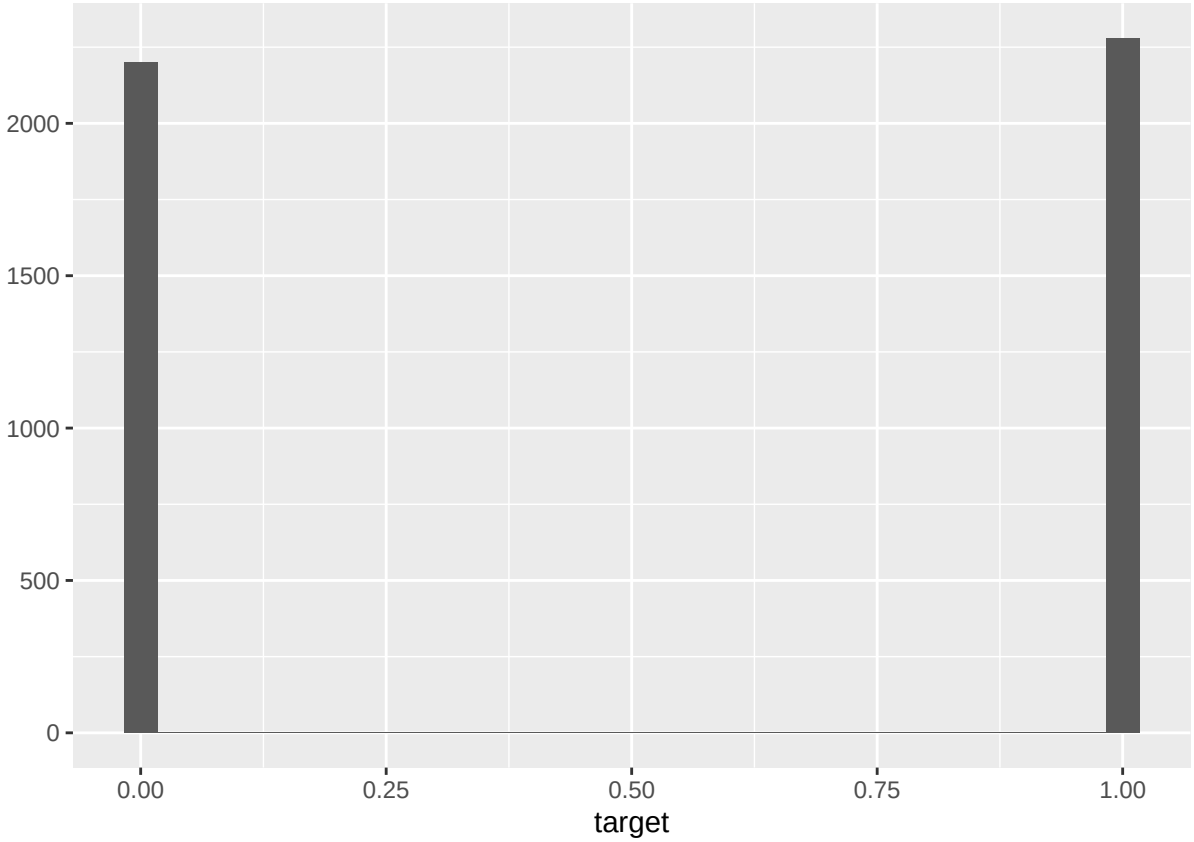
We then split the data into training and test sets, with 30% of the entries being put aside into the test set, and the rest being used for training.

The number of observations in the training set ends up being 4479, with 1919 observations in the test set.

A sample of the data is shown below.

```
## # A tibble: 6 x 9
##   danceability energy loudness speechiness acousticness valence tempo
##          <dbl>  <dbl>    <dbl>       <dbl>        <dbl>   <dbl> <dbl>
## 1        0.741  0.626    -4.83      0.0886      0.02      0.706 108.
## 2        0.447  0.247   -14.7       0.0346      0.871     0.25  155.
## 3        0.55   0.415    -6.56      0.052       0.161     0.274 172.
## 4        0.502  0.648    -5.70      0.0527      0.00513   0.291  91.8
## 5        0.807  0.887    -3.89      0.275       0.00381   0.78  161.
## 6        0.482  0.873    -3.14      0.0853      0.0111    0.737 165.
## # ... with 2 more variables: duration_ms <int>, target <int>
```

```
## Warning: Ignoring unknown parameters: title
```



We see from this graph that the creator of the dataset intended the dataset to be split roughly 50/50 between hits and not hits, which doesn't affect our ability to analyze the data, but may lead to inaccurate results in relation to songs on a larger scale.

Classifiers

We used three methods to develop our classifier. We used two basic classifier methods: logisitic regression and decision trees, and random forests as our ensemble method. Model selection is performed in the next section, and involved confusion matrixes, misclassification rates, and ROC curves to assess overall performances of our models.

Model Building

We used three methods to develop our classifier, two basic classifier methods and an ensemble method, each shown in more detail in the following sections.

Logistic Regression

We fit a logistic regression model to our data using 'target' as the response variable. Through our summary of the model we found that all of the variables that we fit had a statsitically significant effect on 'target' except for valence. A summary of the regression model is shown below.

```
##
## Call:
```

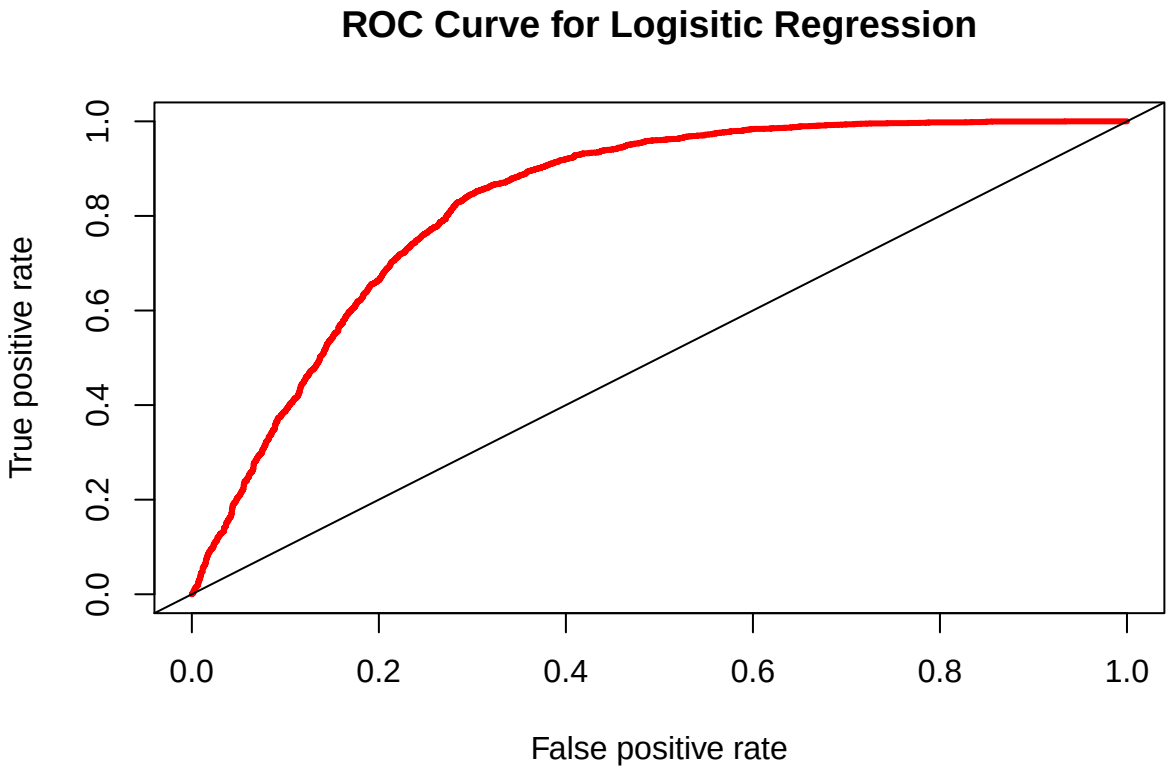
```
## glm(formula = target ~ ., family = binomial, data = music)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9569  -0.8263   0.0714   0.8647   3.3706
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   7.156e+00  3.890e-01  18.394  < 2e-16 ***
## danceability  3.129e+00  2.255e-01  13.873  < 2e-16 ***
## energy       -7.005e+00  2.938e-01 -23.845  < 2e-16 ***
## loudness      4.930e-01  1.802e-02  27.362  < 2e-16 ***
## speechiness   6.836e-01  3.228e-01   2.117   0.0342 *
## acousticness -1.474e+00  1.694e-01  -8.701  < 2e-16 ***
## valence       1.206e-01  1.563e-01   0.772   0.4403
## tempo         2.541e-03  1.068e-03   2.379   0.0173 *
## duration_ms  -3.925e-06  5.001e-07  -7.849  4.2e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8869.5  on 6397  degrees of freedom
## Residual deviance: 6410.8  on 6389  degrees of freedom
## AIC: 6428.8
##
## Number of Fisher Scoring iterations: 6
```

Then we used the predict() function to get a sequence of probabilities and created a confusion matrix for the training data. To do this we picked a threshold value of 0.5 to assign the labels of '0' or '1' for the predicted labels. Then we set up the matrix shown below with the predicted labels and the labels from the actual data. By comparing the predicited values with the true vabels we were able to compute the test error with the equation 1 - sum(diag(matrix))/diag(matrix).

The test error of this model comes out to be .2319475.

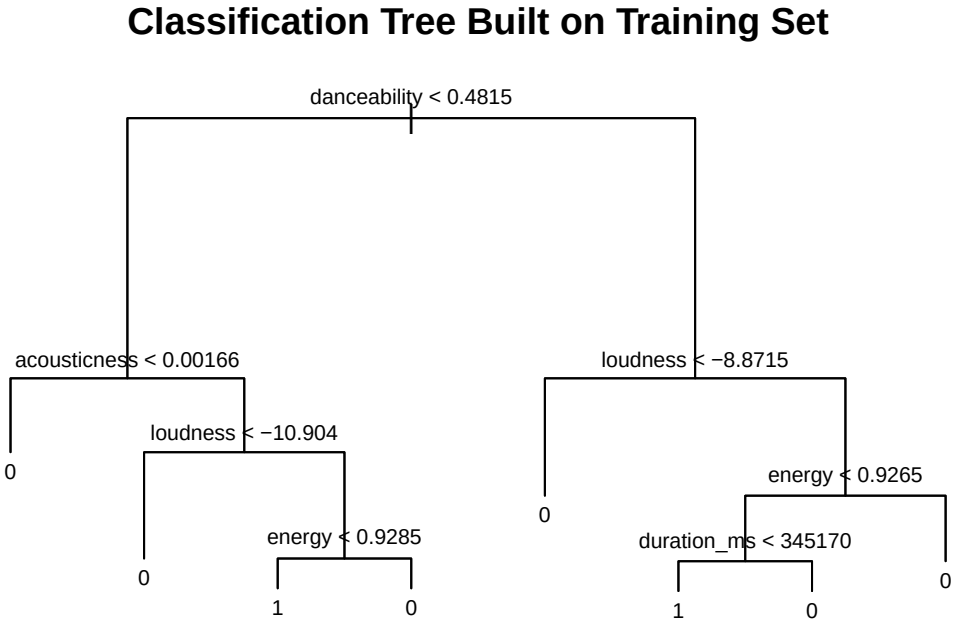
```
##     true
## pred    0    1
##    0 2311  596
##    1  888 2603
```

To further test model adequecy we create a ROC Curve shown below, which plots the true positive rate versus the false positive rate. The area under the curve, or AUC, is used to summarize the overall performance of the model, and is found to be .8284404 which is generally accepted to be a good value.

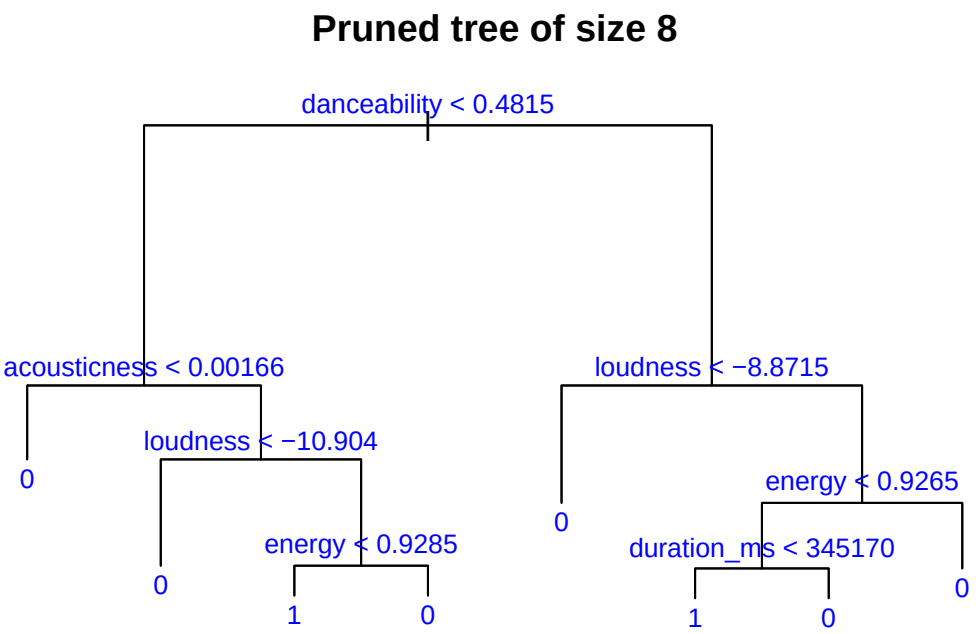


Classification Trees

Another basic classifier method that we attempted was decision trees, which predicts which most commonly occuring region each observation belongs to. Using our training data, we fit the data to a normal decision tree, and the tree shown is in the first figure below.

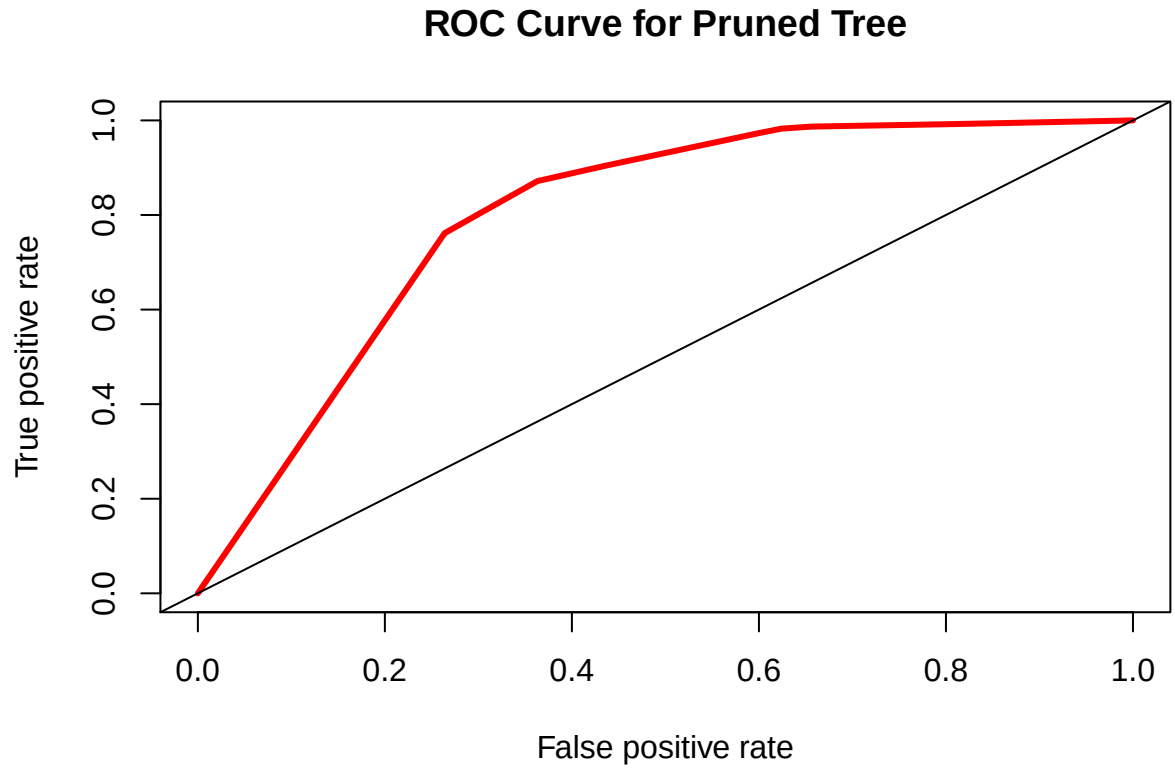


Then we pruned the tree down to a size of 6 and then again based on cross-validation, but all of the plots ended up being the same, suggesting that this method of model building wouldn't provide us with the best results.




```
##         y.test
## cv.pred   0   1
##       0 637 118
##       1 363 801
```

Our classification table consisting of the cross-validated tree predicted values and the test data is used to compute a test error rate of 0.2506514.

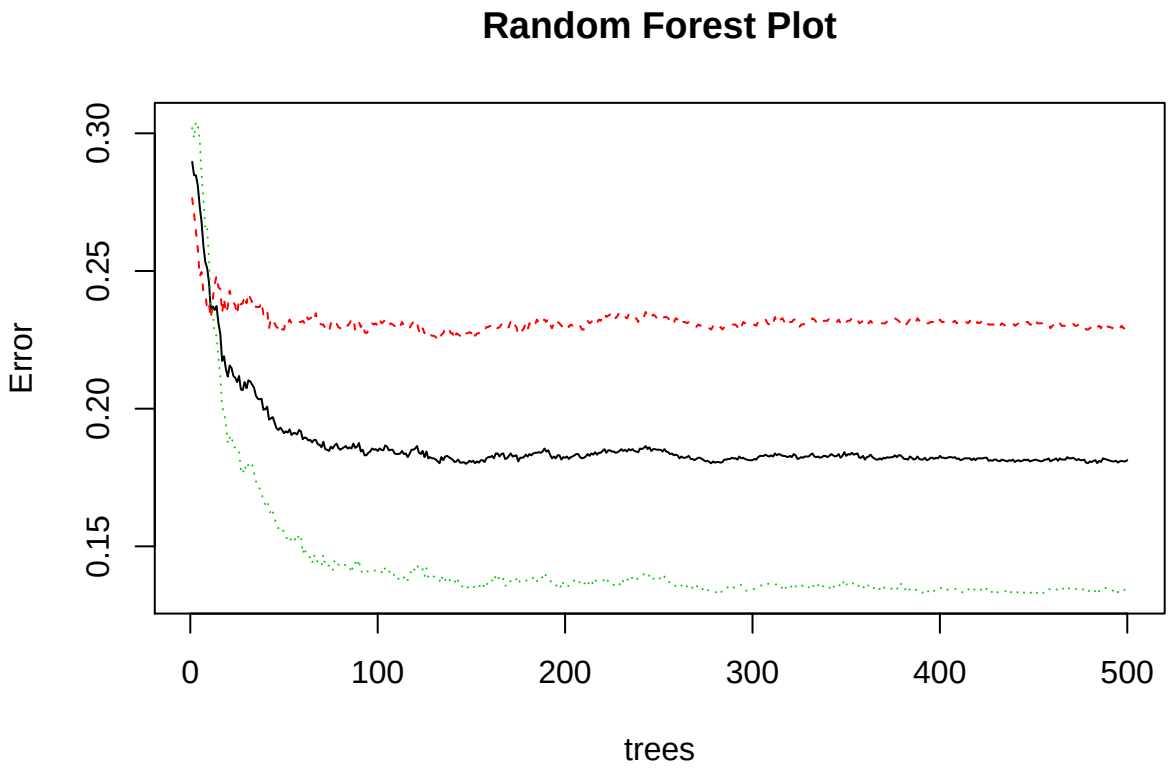


THe ROC curve from the pruned tree is shown above, and the AUC is computed to be 0.796809.

Random Forests

We used random forests as our ensemble learning method of classification, which should be an improvement over our decision trees method since we are a building a large number of decision trees. First we plot the randomForest using mtry=3, since the number of predictors we have is 8, and the number of trees = 500 using the training data. The plotted tree is shown below.

```
##
## Call:
##  randomForest(formula = as.factor(target) ~ ., data = data.train,      mtry = 3, ntree = 500, importa
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 18.13%
## Confusion matrix:
##      0    1 class.error
## 0 1693  506   0.2301046
## 1  306 1974   0.1342105
```



Then we used the importance() and varImpPlot() functions, with their outputs shown below respectively, to examine the randomforest. From this we learned that loudness, energy, and danceability are the most important variables in terms of Model Accuracy. We can also observe that all eight of the features fitted in the model were shown to have some importance, so we deemed it unneccessary to eliminate variables.

```
##                     0         1 MeanDecreaseAccuracy MeanDecreaseGini
## danceability 25.33527  66.41759             70.64510         343.8854
## energy       30.99998  73.40107             82.02818         331.7252
## loudness     48.78717 116.57536            116.77537         414.1852
```

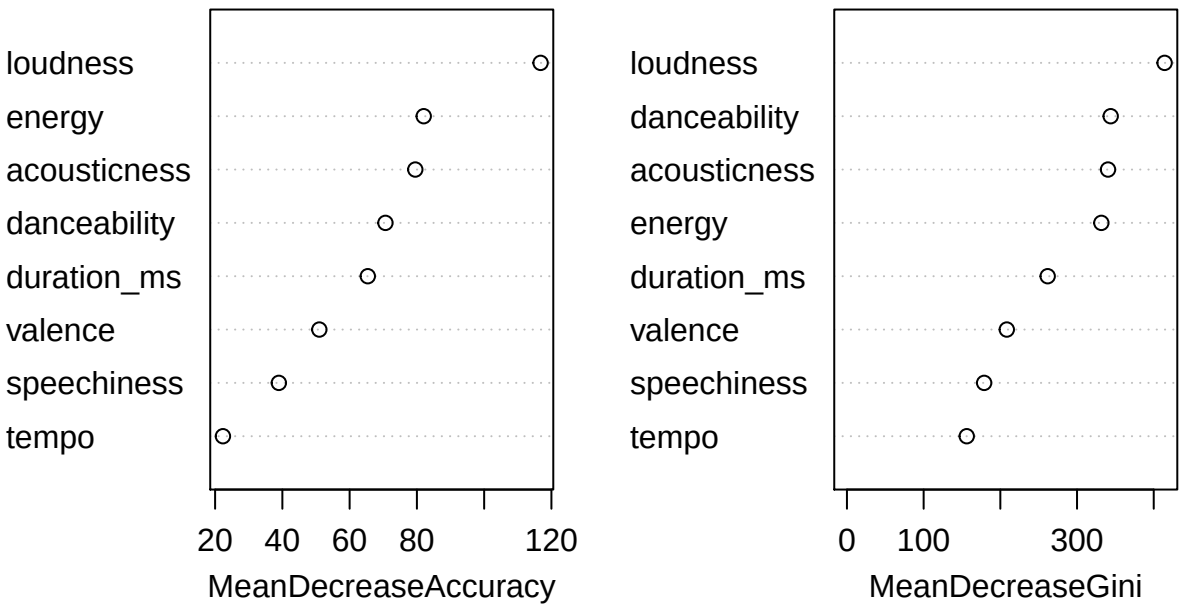
```
## speechiness  34.18055  18.06134            38.95307        178.9073
## acousticness 29.36315  62.94057            79.51994        340.4055
## valence      22.59306  40.10086            50.98887        208.4249
## tempo        12.22027  17.25114            22.32961        156.2005
## duration_ms  21.68283  62.16301            65.39957        261.7274
```

```
## Warning in mtext(labs, side = 2, line = loffset, at = y, adj = 0, col = color, :
## "title" is not a graphical parameter
```

```
## Warning in title(main = main, xlab = xlab, ylab = ylab, ...): "title" is not a
## graphical parameter
```

```
## Warning in mtext(labs, side = 2, line = loffset, at = y, adj = 0, col = color, :
## "title" is not a graphical parameter
```

```
## Warning in title(main = main, xlab = xlab, ylab = ylab, ...): "title" is not a
## graphical parameter
```

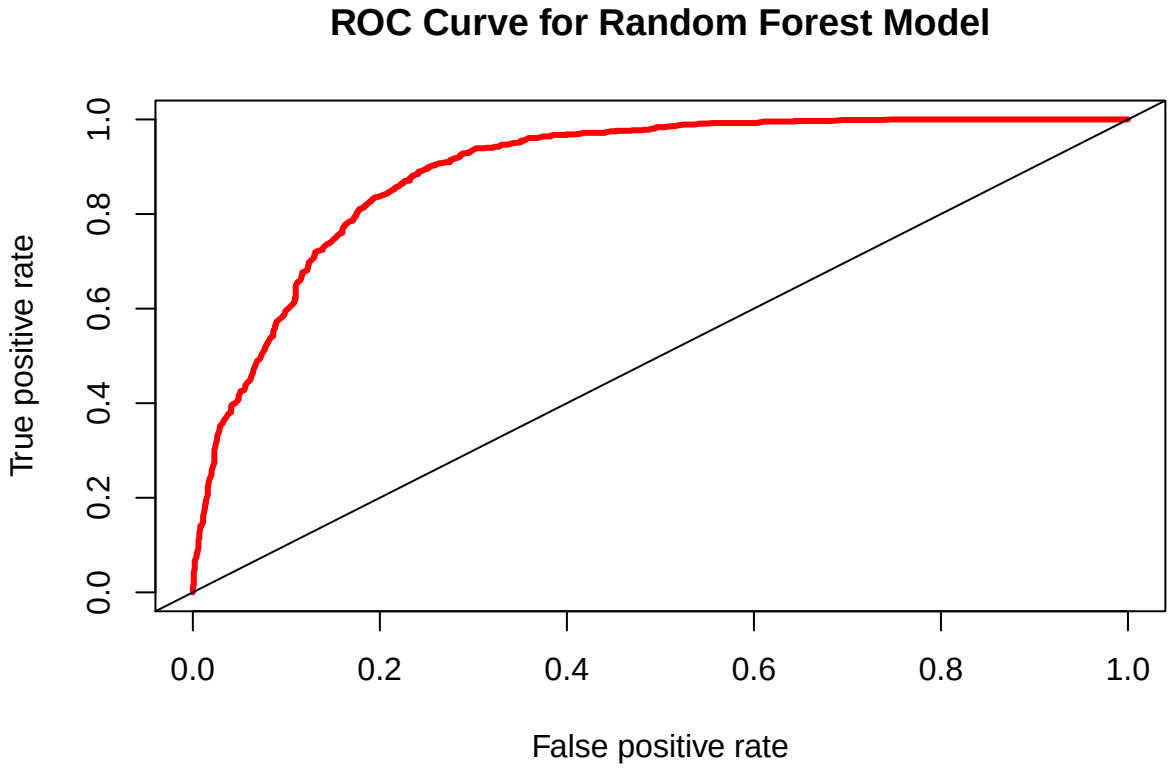



```
##     truth
## pred   0   1
##    0 776 125
##    1 224 794
```

We then obtain the test error rate of 0.1818656 using the confusion matrix shown above by comparing the true values of the test set to the values predicted by the randomforest again with the equation

1 - sum(diag(matrix))/diag(matrix).

Similar to our other models we continue to test model performance with the ROC curve shown below and computing the AUC to be 0.8936442, which is a great value, bordering excellent.



Conclusion

In this project we trained three models in an attempt to develop a classifier that designates whether a song is a hit or not based on various musical features of the song. The three classification methods used were logistic regression, decision trees, and random forests, with random forests yielding the best results.

We looked at two components given from ROC curves and misclassification tables when determining the best model:

Comparing the test errors:

Logistic Regression: 0.2319475

Decision Trees: 0.2506514

Random Forests: 0.1818656

alongside the AUC Values:

Logistic Regression: .8284404

Decision Trees: 0.796809

Random Forests: 0.8936442

From this we determined that the random forests classifier is our best model. In terms of improving the model for further accuracy we considered changing the parameters of the random forest, but any alterations

caused the scores to test error to increase and AUC to decrease. As a result the random forest model we fitted was accepted to be the final model without any changes to the parameters.

For further expansion of this data analysis, one could fit the same models performed here to the other decade datasets and compare results, such as comparing the classifiers from the 70s decade to the most recent 2010s one. Possible inquries would include determining if importance of specific variables changed between decades, or which decades had the most specific trends in terms of musical features. Also, if we wanted to improve the datasets we could pick songs from spotify's database at random, instead of ensuring that we get a 50/50 split between the two 'target' designations.

References

Data Source: https://www.kaggle.com/theoverman/the-spotify-hit-predictor-dataset

All data manipulation and processing done using R and RStudio.