Classification

Kathryn Kingsley KLK170230

In this notebook, I explore a data set regarding Spotify's top songs using logistic regression, Naive Bayes, and decision trees- random forest. This CSV has 174,389 observations with 19 variables.Data can be found here:

`https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks?select=data.csv`

I want to see if the popularity of a song can be predicted.

First I import my data.

```
df <- data.frame(read.csv("music_data.csv"))
```

Clean the data. I remove all unnecessary columns- ones that I felt were irrelevant for the project- and checked for NAs. Since we have so much data,
I simply remove them rather than subbing with the mean in order to not dilute the data.

```
# remove irrelevant columns
df <- df[ , c("acousticness", "danceability", "energy", "explicit",
              "instrumentalness", "liveness", "loudness", "popularity",
              "speechiness", "tempo", "valence", "year")]

# remove instances with NAs or nulls
df <- df[!(df$acousticness=="" | df$danceability==""| df$energy=="" |
             df$explicit==""| df$instrumentalness==""|df$liveness=="" |
             df$loudness==""| df$popularity=="" | df$speechiness==""|
             df$tempo=="" | df$valence==""| df$year==""),]

# ensure all are numeric so I can work with them

df$acousticness <- as.numeric(df$acousticness)
df$danceability <- as.numeric(df$danceability)
df$energy <- as.numeric(df$energy)
df$explicit <- as.numeric(df$explicit)
df$instrumentalness <- as.numeric(df$instrumentalness)
df$liveness <- as.numeric(df$liveness)
df$loudness <- as.numeric(df$loudness)
df$popularity <- as.numeric(df$popularity)
df$speechiness <- as.numeric(df$speechiness)
df$tempo <- as.numeric(df$tempo)
df$valence <- as.numeric(df$valence)
df$year <- as.numeric(df$year)
```

I first did some basic exploration to see what type of variables I was going to be working with. I also created a correlation table with my variables to find those with the strongest relationships.

```
summary(df)
```

```
##   acousticness     danceability        energy          explicit
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0877   1st Qu.:0.4140   1st Qu.:0.2490   1st Qu.:0.00000
##  Median :0.5170   Median :0.5480   Median :0.4650   Median :0.00000
##  Mean   :0.4992   Mean   :0.5368   Mean   :0.4827   Mean   :0.06814
##  3rd Qu.:0.8950   3rd Qu.:0.6690   3rd Qu.:0.7110   3rd Qu.:0.00000
##  Max.   :0.9960   Max.   :0.9880   Max.   :1.0000   Max.   :1.00000
##  instrumentalness      liveness         loudness         popularity
##  Min.   :0.000000   Min.   :0.0000   Min.   :-60.000   Min.   :  0.00
##  1st Qu.:0.000000   1st Qu.:0.0992   1st Qu.:-14.908   1st Qu.:  1.00
##  Median :0.000524   Median :0.1380   Median :-10.836   Median : 25.00
##  Mean   :0.197252   Mean   :0.2111   Mean   :-11.751   Mean   : 25.69
##  3rd Qu.:0.252000   3rd Qu.:0.2700   3rd Qu.: -7.499   3rd Qu.: 42.00
##  Max.   :1.000000   Max.   :1.0000   Max.   :  3.855   Max.   :100.00
##   speechiness         tempo           valence           year
##  Min.   :0.0000   Min.   :  0.00   Min.   :0.0000   Min.   :1920
##  1st Qu.:0.0352   1st Qu.: 93.93   1st Qu.:0.3110   1st Qu.:1955
##  Median :0.0455   Median :115.82   Median :0.5360   Median :1977
##  Mean   :0.1057   Mean   :117.01   Mean   :0.5245   Mean   :1977
##  3rd Qu.:0.0763   3rd Qu.:135.01   3rd Qu.:0.7430   3rd Qu.:1999
##  Max.   :0.9710   Max.   :243.51   Max.   :1.0000   Max.   :2021
```

```
names(df)
```

```
##  [1] "acousticness"     "danceability"     "energy"           "explicit"
##  [5] "instrumentalness" "liveness"         "loudness"         "popularity"
##  [9] "speechiness"      "tempo"            "valence"          "year"
```

```
head(df)
```

```
##   acousticness danceability energy explicit instrumentalness liveness loudness
## 1     0.991000        0.598  0.224        0         5.22e-04   0.3790  -12.628
## 2     0.643000        0.852  0.517        0         2.64e-02   0.0809   -7.261
## 3     0.993000        0.647  0.186        0         1.76e-05   0.5190  -12.098
## 4     0.000173        0.730  0.798        0         8.01e-01   0.1280   -7.311
## 5     0.295000        0.704  0.707        1         2.46e-04   0.4020   -6.036
## 6     0.996000        0.424  0.245        0         7.99e-01   0.2350  -11.470
##   popularity speechiness   tempo valence year
## 1         12      0.0936 149.976  0.6340 1920
## 2          7      0.0534  86.889  0.9500 1920
## 3          4      0.1740  97.600  0.6890 1920
## 4         17      0.0425 127.997  0.0422 1920
## 5          2      0.0768 122.076  0.2990 1920
## 6          9      0.0397 103.870  0.4770 1920
```

```
tail(df)
```

```
##        acousticness danceability energy explicit instrumentalness liveness
## 174384      0.79500        0.429  0.211        0         0.00e+00    0.196
## 174385      0.00917        0.792  0.866        0         5.99e-05    0.178
## 174386      0.79500        0.429  0.211        0         0.00e+00    0.196
## 174387      0.80600        0.671  0.589        0         9.20e-01    0.113
## 174388      0.92000        0.462  0.240        1         0.00e+00    0.113
## 174389      0.23900        0.677  0.460        0         8.91e-01    0.215
##        loudness popularity speechiness   tempo valence year
## 174384  -11.665          0      0.0360  94.710   0.228 2021
## 174385   -5.089          0      0.0356 125.972   0.186 2020
## 174386  -11.665          0      0.0360  94.710   0.228 2021
## 174387  -12.393          0      0.0282 108.058   0.714 2020
## 174388  -12.077         69      0.0377 171.319   0.320 2021
## 174389  -12.237          0      0.0258 112.208   0.747 2020
```

```
dim(df)
```

```
## [1] 174389     12
```

```
range(df$year)
```

```
## [1] 1920 2021
```

```
range(df$popularity)
```

```
## [1]   0 100
```

```
cor(df, use="complete")
```

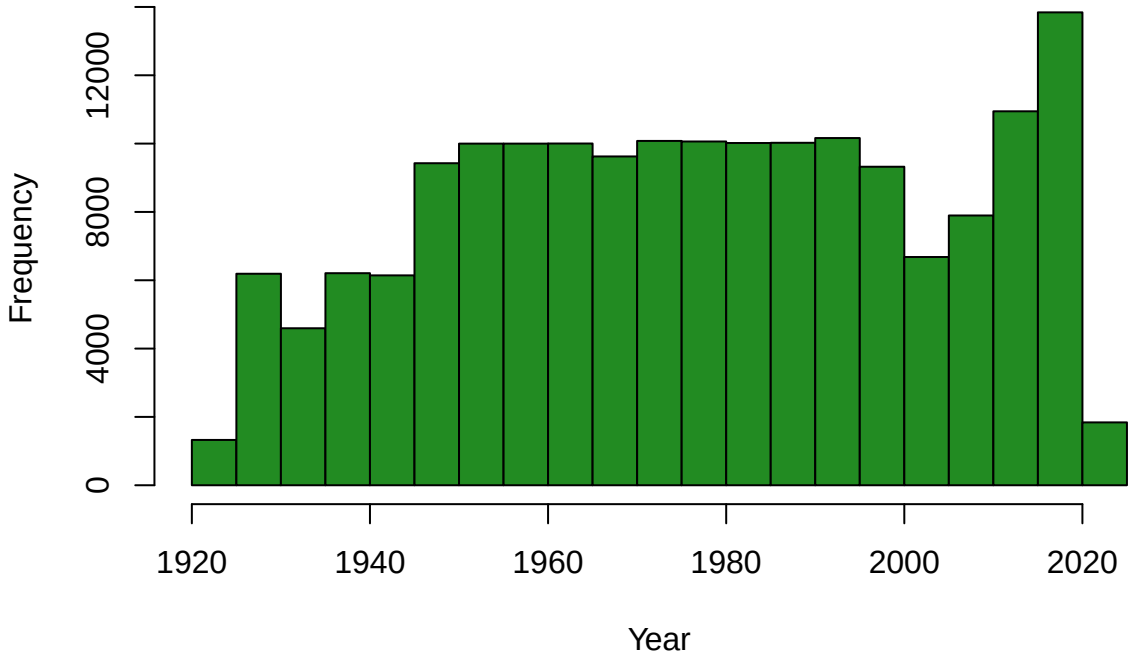
```
##                  acousticness danceability     energy     explicit
## acousticness       1.00000000 -0.263217324 -0.7508523 -0.208175897
## danceability      -0.26321732  1.000000000  0.2048379  0.200842313
## energy            -0.75085230  0.204837910  1.0000000  0.102560856
## explicit          -0.20817590  0.200842313  0.1025609  1.000000000
## instrumentalness   0.22195611 -0.215588948 -0.1777502 -0.130609400
## liveness          -0.02965412 -0.110033207  0.1348148  0.037287947
## loudness          -0.54663949  0.249541036  0.7792669  0.106249419
## popularity        -0.39674405  0.123745802  0.3289386  0.152545482
## speechiness       -0.02243747  0.239962010 -0.1126165  0.353871889
## tempo             -0.22383963  0.005478628  0.2664475  0.008075114
## valence           -0.16696761  0.536712663  0.3264185 -0.009275059
## year              -0.60751549  0.159094698  0.5408497  0.151619475
##                  instrumentalness     liveness    loudness  popularity
## acousticness           0.22195611 -0.029654123 -0.54663949 -0.39674405
## danceability          -0.21558895 -0.110033207  0.24954104  0.12374580
## energy                -0.17775023  0.134814844  0.77926685  0.32893862
## explicit              -0.13060940  0.037287947  0.10624942  0.15254548
## instrumentalness       1.00000000 -0.047941114 -0.31756244 -0.30062511
```

```
## liveness              -0.04794111  1.000000000  0.06269508 -0.07895912
## loudness              -0.31756244  0.062695076  1.00000000  0.33719411
## popularity            -0.30062511 -0.078959117  0.33719411  1.00000000
## speechiness           -0.13396563  0.122033676 -0.21350425 -0.19532890
## tempo                 -0.06865639  0.008585515  0.21791362  0.09498458
## valence               -0.21918767 -0.005780708  0.30251953  0.06347111
## year                  -0.11425939 -0.011851892  0.46518863  0.51322680
##                   speechiness        tempo      valence        year
## acousticness     -0.02243747 -0.223839631 -0.166967611 -0.60751549
## danceability      0.23996201  0.005478628  0.536712663  0.15909470
## energy           -0.11261648  0.266447519  0.326418493  0.54084966
## explicit          0.35387189  0.008075114 -0.009275059  0.15161948
## instrumentalness -0.13396563 -0.068656392 -0.219187669 -0.11425939
## liveness          0.12203368  0.008585515 -0.005780708 -0.01185189
## loudness         -0.21350425  0.217913622  0.302519530  0.46518863
## popularity       -0.19532890  0.094984577  0.063471111  0.51322680
## speechiness       1.00000000 -0.033530030  0.050600141 -0.21563033
## tempo            -0.03353003  1.000000000  0.163117771  0.16172881
## valence           0.05060014  0.163117771  1.000000000 -0.04957847
## year             -0.21563033  0.161728809 -0.049578466  1.00000000
```

Create some graphs so we can see a visual representation of the data. I first want to see the spread for the years- how many songs from each era there are. Then, I'm curious about the spread of popularity. Do we have a lot of popular songs to work with in this giant data set? Lastly, I want to see popularity as a function of year.

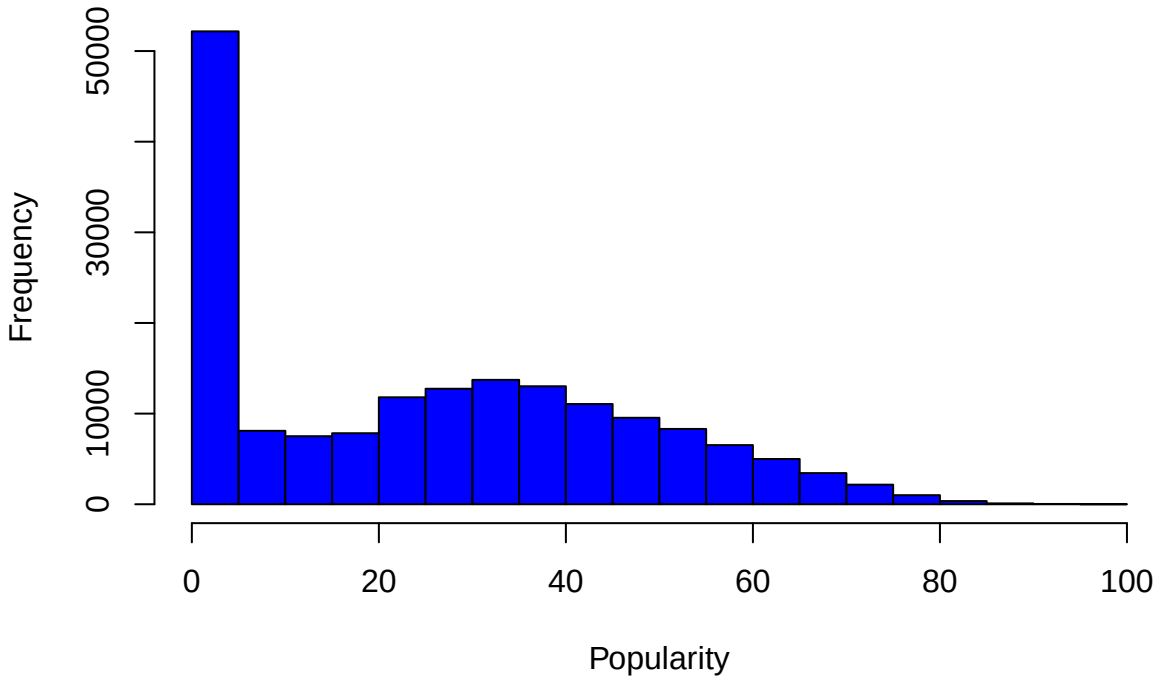
```
hist(df$year, col="forest green", main="Figure 2.1",xlab="Year")
```

Figure 2.1



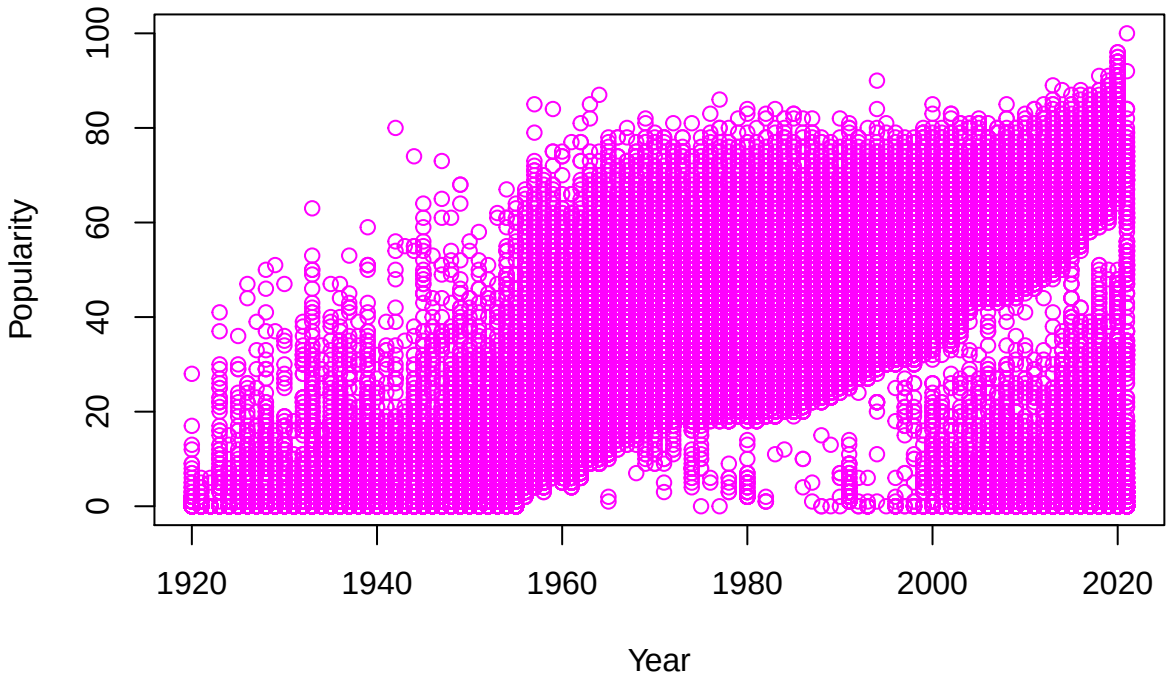

```
hist(df$popularity, col="blue", main="Figure 2.2",xlab="Popularity")
```

Figure 2.2




```
plot(df$popularity~ df$year, main="Figure 2.3", col="magenta", xlab="Year",
     ylab="Popularity")
```

Figure 2.3



As you can see, the middle years are spread pretty evenly throughout the sampling with more recent music having more, and really old music having less. There are way more unpopular samples, and limited really popular samples. So this isn't looking good. Clearly, more recent music tends to be more popular as well, which is probably generational bias for Spotify users. Yikes. I may have picked a bad data set, but there does seem to be some correlation between year and popularity. I hope this saves my results.

Next I create a column is.Popular to create binary categories- >50 is considered popular and then separate into train and test sets. 75% train and 25% test.

```
df$is.Popular <- as.factor(ifelse(df$popularity >= 50,1 ,0))
set.seed(1234)
i <- sample(1:nrow(df), nrow(df)*0.75, replace = FALSE)
train <- df[i,]
test <- df[-i,]
```

Logistic Regression Create a logistic regression model where is.Popular is a function of year, energy, and acousticness. I picked these because year was those most related factor with popularity, and year correlated highly with energy and acousticness.

```
start_time1 <- Sys.time()
glm1 <- glm(is.Popular~energy+acousticness+year ,data=train, family="binomial")
end_time1 <- Sys.time()
final_time1 <- end_time1 - start_time1
summary(glm1)
```

```
##
## Call:
## glm(formula = is.Popular ~ energy + acousticness + year, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2735  -0.5993  -0.3154  -0.1702   3.1024
##
## Coefficients:
##                Estimate Std. Error  z value Pr(>|z|)
## (Intercept)  -9.849e+01  8.960e-01 -109.923  < 2e-16 ***
## energy       -1.609e-01  4.726e-02   -3.405 0.000662 ***
## acousticness -4.763e-01  3.681e-02  -12.939  < 2e-16 ***
## year          4.886e-02  4.487e-04  108.908  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 117101  on 130790  degrees of freedom
## Residual deviance:  94134  on 130787  degrees of freedom
## AIC: 94142
##
## Number of Fisher Scoring iterations: 6
```

The model looks decent with low p-values.

Now I predict on the test data, and use "response" to return a vector or probabilities.

```
probs1<- predict(glm1, newdata=test, type="response")
pred1 <- ifelse(probs1>0.5, 1, 0)
```

Calculate and output accuracy, sensitive, specificity, and time.

```
acc1 <- mean(pred1==test$is.Popular)
calc_table1 <- table(pred1, test$is.Popular)
calc_table1
```

```
##
## pred1     0     1
##     0 35316  6821
##     1  1147   314
```

```
tp1 <- calc_table1[1]
fn1 <- calc_table1[2]
fp1 <- calc_table1[3]
tn1 <- calc_table1[4]
sens1 <- (tp1/(tp1+fn1))
spec1 <- (tn1/(tn1+fp1))

# print out
print(paste("Time taken LR1 : ", final_time1))
```

```
## [1] "Time taken LR1 :  0.795871019363403"
```

```
print(paste("Accuracy LR1: ", acc1))
```

```
## [1] "Accuracy LR1:  0.81723932290472"
```

```
print(paste("Specificity LR1: ", spec1))
```

```
## [1] "Specificity LR1:  0.0440084092501752"
```

```
print(paste("Sensitivity LR1: ", sens1))
```

```
## [1] "Sensitivity LR1:  0.968543455009187"
```

I want to see if I can get a little higher than 81% accuracy, so I'm going to add daceability and energy, because I personally think this will impact popularity.

```
start_time1 <- Sys.time()
glm1 <- glm(is.Popular~danceability+energy+acousticness+loudness+year ,
            data=train, family="binomial")
end_time1 <- Sys.time()
final_time1 <- end_time1 - start_time1
summary(glm1)
```

```
##
## Call:
## glm(formula = is.Popular ~ danceability + energy + acousticness +
##     loudness + year, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7344  -0.5535  -0.3191  -0.1534   4.0703
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -8.744e+01  9.276e-01 -94.270   <2e-16 ***
## danceability  5.023e-01  5.080e-02   9.889   <2e-16 ***
## energy       -2.350e+00  6.429e-02 -36.559   <2e-16 ***
## acousticness -4.659e-01  3.828e-02 -12.171   <2e-16 ***
## loudness      1.644e-01  3.146e-03  52.256   <2e-16 ***
## year          4.461e-02  4.615e-04  96.658   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 117101  on 130790  degrees of freedom
## Residual deviance:  90653  on 130785  degrees of freedom
## AIC: 90665
##
## Number of Fisher Scoring iterations: 6
```

The model looks good again. Let's see about accuracy.

```
probs1<- predict(glm1, newdata=test, type="response")
pred1 <- ifelse(probs1>0.5, 1, 0)
acc1 <- mean(pred1==test$is.Popular)
calc_table1 <- table(pred1, test$is.Popular)
calc_table1
```

```
##
## pred1     0     1
##     0 35090  5534
##     1  1373  1601
```

```
tp1 <- calc_table1[1]
fn1 <- calc_table1[2]
fp1 <- calc_table1[3]
tn1 <- calc_table1[4]
sens1 <- (tp1/(tp1+fn1))
spec1 <- (tn1/(tn1+fp1))

# print out
print(paste("Time taken LR2 : ", final_time1))
```

```
## [1] "Time taken LR2 :  0.850725889205933"
```

```
print(paste("Accuracy LR2: ", acc1))
```

```
## [1] "Accuracy LR2:  0.84157530161934"
```

```
print(paste("Specificity LR2: ", spec1))
```

```
## [1] "Specificity LR2:  0.224386825508059"
```

```
print(paste("Sensitivity LR2: ", sens1))
```

```
## [1] "Sensitivity LR2:  0.962345391218495"
```

It actually did have slightly better results with more dimensions at 84% accuracy.

Naive Bayes

Turn all continuous variables into multilevel factors, and create new test and train set

```
df2 <- df  # store in its own var so we don't mess up first data frame
df2$year <- findInterval(df2$year,  c(1920, 1960))

set.seed(1234)
i <- sample(1:nrow(df2), nrow(df2)*0.75, replace = FALSE)
train2 <- df2[i,]
test2 <- df2[-i,]
```

Create Naive Bayes model on train data. I used the same predictors as log reg 1 - year, energy, and acousticness

```
library(e1071) # required for Naive Bayes
start_time2 <- Sys.time()
nb1 <- naiveBayes(is.Popular~as.factor(year)+energy+acousticness,
                  data = train2)
end_time2 <- Sys.time()
final_time2 <- end_time2 - start_time2
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.8351263 0.1648737
##
## Conditional probabilities:
##    as.factor(year)
## Y            1          2
##   0 0.35433547 0.64566453
##   1 0.01061955 0.98938045
##
##    energy
## Y        [,1]      [,2]
##   0 0.4568322 0.2728917
##   1 0.6158297 0.2294366
##
##    acousticness
## Y        [,1]      [,2]
##   0 0.5411371 0.3797516
##   1 0.2836707 0.2985509
```

This model doesn't look as good since the A-priori's are so heavily skewed toward unpopular. Let's continue though.

Test on test data.

```
pred2 <- predict(nb1, newdata = test2, type= "class")
```

Calculate and output information

```
calc_table2<- table(pred2, test2$is.Popular)
acc2 <- mean(pred2==test2$is.Popular)

# print out
print(paste("Time taken: ", final_time2))
```

```
## [1] "Time taken:  0.17357611656189"
```

```
print(paste("Accuracy: ", acc2))
```

```
## [1] "Accuracy:  0.836345703931373"
```

Not as bad as I anticipated. It is nearly as accurate as logistic regression 2, but I don't know if that is true accuracy, or if the fact that there is an 86% likelihood that a song is unpopular is skewing the results.

Decision Trees Since I had such poor luck with Log Regression and Naive Bayes, I was hoping a decision tree might be able to help. Here, I separated the 100 different levels for popularity into 2. Since there were so many unpopular samples, anything<= 10 I called "unpopular" and larger than that, I'm classifying as popular.

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.0.4
```

```
set.seed(1234)
df3 <- df
# df3$popularity <- findInterval(df3$popularity,  c(0,10,20,30,40,50,60,70,80,90))

for (x in 1:nrow(df3)){

      if(df3$popularity[x]<=60){
        df3$popularity[x]<-1
      }
      else if(df3$popularity[x]<=100){
        df3$popularity[x]<-2
      }

}
```

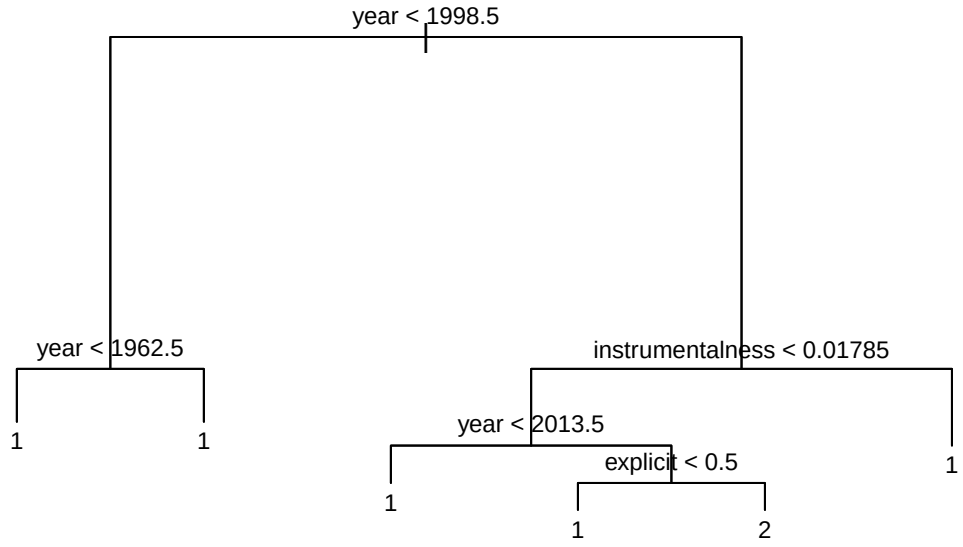
Create my tree

```
train3 <- df3[i,]
test3 <- df3[-i,]
start_time3 <- Sys.time()
tree1 <- tree(as.factor(popularity)~.-is.Popular, data=train3)
end_time3 <- Sys.time()
final_time3 <- end_time3 - start_time3
summary(tree1)
```

```
##
## Classification tree:
## tree(formula = as.factor(popularity) ~ . - is.Popular, data = train3)
## Variables actually used in tree construction:
## [1] "year"             "instrumentalness" "explicit"
## Number of terminal nodes:  6
## Residual mean deviance:  0.3672 = 48020 / 130800
## Misclassification error rate: 0.06302 = 8243 / 130791
```

Make visualization of the tree

```
plot(tree1)
text(tree1, cex=0.75, pretty=0)
```



Predict on test data

```
pred3 <- predict(tree1, newdata=test3, type="class")
```

Calculate and output results

```
calc_table3<- table(pred3, test3$popularity)
acc3 <- mean(pred3==test3$popularity)
calc_table3
```

```
##
## pred3     1     2
##     1 40311  2394
##     2   305   588
```

```
# print out
print(paste("Time taken DT1: ", final_time3))
```

```
## [1] "Time taken DT1:  1.44105887413025"
```

```
print(paste("Accuracy DT1: ", acc3))
```

```
## [1] "Accuracy DT1:  0.938093490527088"
```

This is the best yet! almost 94% accuracy, but it was a bit slower.

Testing Decision Tree I just want to see if I slice the data a bit better, if I'd have gotten better results. Since the first split in the tree was at year 1955, and looking at my graph (Figure 2.3), the split from popular to unpopular appears more defined from about year 1990, I'm going to only take observations from 1990 and onward. Since 80 is about as high as it goes for the majority of the graph 3, I set the 3 categories as low (1) being anything lower than a 25, mid (2) is anything lower than 50,and high popularity (3) as anything else.

```
# df$acousticness <- as.numeric(df$acousticness)
# df$danceability <- as.numeric(df$danceability)
# df$energy <- as.numeric(df$energy)
# df$explicit <- as.numeric(df$explicit)
# df$instrumentalness <- as.numeric(df$instrumentalness)
# df$liveness <- as.numeric(df$liveness)
# df$loudness <- as.numeric(df$loudness)
# df$popularity <- as.numeric(df$popularity)
# df$speechiness <- as.numeric(df$speechiness)
# df$tempo <- as.numeric(df$tempo)
# df$valence <- as.numeric(df$valence)
# df$year <- as.numeric(df$year)
# get rid of old songs
df4 <- df[!( df$year<1990),]

# gave it its own loop in case i wanted to play with proportions
for (x in 1:nrow(df4)){

      if(df4$popularity[x]<=60){
        df4$popularity[x]<-1
      }
      else if(df4$popularity[x]<=100){
        df4$popularity[x]<-2
      }
}

# create tree

train3 <- df4[i,]
test3 <- df4[-i,]
start_time3 <- Sys.time()
tree2 <- tree(as.factor(popularity)~.-is.Popular, data=train3)
end_time3 <- Sys.time()
final_time3 <- end_time3 - start_time3
summary(tree2)
```

```
##
## Classification tree:
## tree(formula = as.factor(popularity) ~ . - is.Popular, data = train3)
## Variables actually used in tree construction:
## [1] "instrumentalness" "year"             "explicit"
```

```
## Number of terminal nodes:  5
## Residual mean deviance:  0.7528 = 35480 / 47130
## Misclassification error rate: 0.1504 = 7092 / 47139
```

Predict on this new testing tree

```
pred4 <- predict(tree2, newdata=test3, type="class")
```

Output metrics.

```
calc_table3<- table(pred4, test3$popularity)
acc3 <- mean(pred4==test3$popularity)
calc_table3
```

```
##
## pred4     1     2
##     1 12544  1970
##     2   415   631
```

```
# print out
print(paste("Time taken DT2: ", final_time3))
```

```
## [1] "Time taken DT2:  0.502126932144165"
```

```
print(paste("Accuracy DT2: ", acc3))
```

```
## [1] "Accuracy DT2:  0.84672236503856"
```

Well, that was a bust. DT performed better with more data even thought the popularity split was in the same spot.

Results analysis

The best algorithm, hands down, was Decision trees. It was slightly slower, but in the grand scheme of things, a whopping whole second is nothing to sacrifice for about a 10% increase in accuracy. Not to mention, it was very easy to implement.

Second best was logistic regression. Logistic regression had good accuracy at 84% and was extremely fast due to how inexpensive the algorithm is! It was also easy to implement, but I couldn't get the accuracy where I wanted it after some playing around with predictors, likely because my data wasn't linearly separable.

I think both logistic regression and Naive Bayes suffered from the size of the data set, as they typically do better with smaller data sets, but Naive Bayes did the worst of all. It simply could not handle the size of the data. I was number 2 in speed, but that accuracy left a lot to be desired.

Cleaning the data was definitely the hardest part of Classification. Getting the data in workable order is, well, work. Once the cleaning was done, all of these algorithms were relatively easy to implement. I had originally thought there would be no way to predict popular songs as music tastes are so relative, and the visual data of my graphs freaked me out. I was very pleasantly surprised to get up to 94% accuracy with Decision Trees for something that varies so greatly from person to person.