Decoding the Musical Genome

Lakshya Dugar

6/18/2020

Contents

INTRODUCTION

Google defines music as a collection of coordinated sounds. Making music is said to be the process of putting sounds and tones in an order.

In the first part of this project I will try to determine if music can be decoded and if every genre can be explained and appropriately predicted in its simplest form, using the following data-points:

Danceability - the ease with which a person could dance to a song over the course of the whole song,
Energy - how fast paced vs slow paced the song is,
Key - the major or minor scale around which a piece of music revolves,
Loudness - attribute of auditory sensation in terms of which sounds can be ordered on a scale extending from quiet to loud,
Mode - a type of musical scale coupled with a set of characteristic melodic behaviors,
Speechness - the presence of spoken words in a track,
Acousticness - describes how acoustic a song is,
Instrumentalness - the amount of vocals in the song,
Liveness - probability that the song was recorded with a live audience,
Valence - the musical positiveness conveyed by a track,
Tempo - the pace or speed at which a section of music is played (BPM),
Duration_ms - duration of the song in minutes,
time_signature - Release date.

We will try to predict the genres just from these criterion and see if genres are essentially just their genome or is there something else. Additionally, since this data set of just 131,580 songs has 626 genres, we will see what is the state of the overlap and if just using the genome is viable to predict the genres.

Aim - To see analysing just the musical genome is a viable model for the predictive analysis of the genres.

LOADING LIBRARIES

```
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(caTools)) install.packages("caTools", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(class)) install.packages("class", repos = "http://cran.us.r-project.org")
if(!require(mlbench)) install.packages("mlbench", repos = "http://cran.us.r-project.org")
if(!require(glmnet)) install.packages("glmnet", repos = "http://cran.us.r-project.org")
if(!require(nnet)) install.packages("nnet", repos = "http://cran.us.r-project.org")
if(!require(DescTools)) install.packages("DescTools", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
```

DATA WRANGLING

To complete this classification exercise, we're going to borrow **Adri Molina's dataset** from Kaggle which contains a TSV file full of songs and features that will help us categorize the songs into groups (like time signature , key, and tempo).

Let's be sure to eliminate any columns that aren't useful features, and change any factors (besides the predicted factor, "Genre") to numerics, to simplify things

```
#Read in the file, which is tab-delimited
data.full <-
  read.delim("songDb.tsv", header = TRUE, sep = "\t")

# Remove columns that don't serve as features (the Spotify URI,
# The track reference, the full URL, etc.)
datax <-
  subset(data.full,
         select = -c(Uri, Ref_Track, URL_features, Type, ID, Name))

# Identify each song by its name (by changing the row names to song names)
rownames(datax) <- make.names(data.full$Name, unique = TRUE)
# Ensure the time signature is numeric, rather than a factor
datax$time_signature <- as.numeric(datax$time_signature)

# Tempo should also be numeric
datax$Tempo <- as.numeric(datax$Tempo)
```

Now the data is ready in the format that is suitable for our analysis and we can proceed with the project.

This is what our data set looks like:

```
as_tibble(datax)
```

```
## # A tibble: 131,580 x 14
##    Danceability Energy   Key Loudness  Mode Speechness Acousticness
##           <dbl>  <dbl> <dbl>    <dbl> <dbl>      <dbl>        <dbl>
##  1        0.624  0.857    10    -6.25     0     0.0542    0.0208
##  2        0.517  0.916     0    -4.93     1     0.0559    0.000182
##  3        0.251  0.894     8    -4.10     0     0.057     0.0144
##  4        0.469  0.743     1    -5.57     0     0.0272    0.00222
##  5        0.487  0.952     1    -4.43     0     0.0613    0.000228
##  6        0.43   0.797     2    -5.91     0     0.0303    0.000308
##  7        0.434  0.908     6    -4.72     1     0.0936    0.00791
##  8        0.308  0.965     8    -3.17     1     0.0591    0.0000228
##  9        0.5    0.925     4    -3.47     0     0.0378    0.00094
## 10        0.479  0.977     2    -4.51     1     0.086     0.0000166
## # ... with 131,570 more rows, and 7 more variables: Instrumentalness <dbl>,
## #   Liveness <dbl>, Valence <dbl>, Tempo <dbl>, Duration_ms <dbl>,
## #   time_signature <dbl>, Genre <fct>
```

These are the summary statistics:

```r
summary(datax)
```

```
##   Danceability        Energy            Key            Loudness
##  Min.   :0.0000   Min.   : 0.0000   Min.   :-14.372   Min.   :-60.000
##  1st Qu.:0.4320   1st Qu.: 0.4870   1st Qu.:  2.000   1st Qu.:-10.377
##  Median :0.5660   Median : 0.6900   Median :  5.000   Median : -7.377
##  Mean   :0.5538   Mean   : 0.6488   Mean   :  5.311   Mean   : -8.523
##  3rd Qu.:0.6920   3rd Qu.: 0.8530   3rd Qu.:  9.000   3rd Qu.: -5.344
##  Max.   :0.9880   Max.   :11.0000   Max.   : 11.000   Max.   :  5.056
##
##       Mode         Speechness       Acousticness     Instrumentalness
##  Min.   :0.000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000000
##  1st Qu.:0.000   1st Qu.:0.03590   1st Qu.:0.00793   1st Qu.:0.0000019
##  Median :1.000   Median :0.04830   Median :0.10500   Median :0.0014800
##  Mean   :0.619   Mean   :0.08374   Mean   :0.27099   Mean   :0.2318704
##  3rd Qu.:1.000   3rd Qu.:0.08300   3rd Qu.:0.48700   3rd Qu.:0.5030000
##  Max.   :1.000   Max.   :0.96600   Max.   :0.99600   Max.   :0.9990000
##
##     Liveness         Valence             Tempo         Duration_ms
##  Min.   :0.0000   Min.   :  0.0000   Min.   :    1   Min.   :      3
##  1st Qu.:0.0951   1st Qu.:  0.2470   1st Qu.:11846   1st Qu.: 190933
##  Median :0.1250   Median :  0.4590   Median :22406   Median : 229000
##  Mean   :0.1933   Mean   :  0.4938   Mean   :25051   Mean   : 253666
##  3rd Qu.:0.2460   3rd Qu.:  0.6830   3rd Qu.:39009   3rd Qu.: 285479
##  Max.   :1.0000   Max.   :187.8270   Max.   :54265   Max.   :5949886
##
##  time_signature                     Genre
##  Min.   : 1.000   alternativeamericana:  1891
##  1st Qu.: 4.000   electrolatino       :  1009
##  Median : 4.000   doo-wop             :   972
##  Mean   : 3.914   reading             :   969
##  3rd Qu.: 4.000   nuelectro           :   909
##  Max.   :14.000   groovemetal         :   903
##                   (Other)             :124927
```

FUNCTIONS and METHODS

Let's load the functions that we'll need for the first part of our project

Test and Train Sets

Let's also define a function that makes it easy to create our train and test sets. (Storing both, the full train and test sets as well as the separate X and Y data frames for each might seem redundant but it will make things just a bit easier down the road)

```
get_train_test <- function(split_ratio, data) {
  results <- list()

  split.index <- sample.split(seq_len(nrow(data)), split_ratio)

  results$data.train <- data[split.index, ]
  results$data.test <- data[!split.index, ]

  results$X.train <-
    results$data.train %>% select(-Genre) %>% as.matrix()
  results$Y.train <- results$data.train$Genre

  results$X.test <-
    results$data.test %>% select(-Genre) %>% as.matrix()
  results$Y.test <- results$data.test$Genre
  return(results)
}
```

kNN Model

Let's make a function which allows us the subset the entire dataset to contain songs only of a particular genre and will give us the accuracy of our algorithm using the k-Nearest-Neighbours model. Why we subset our data to include only specific genres will become clear later.

```
knn_function <- function(data, genres) {
  data.sub <- data[data$Genre %in% genres, ]
  data.sub$Genre <- droplevels(data.sub$Genre)

  set.seed(101)
  # Create an empty data frame to store the predictions and the actual labels
  classifications <- data.frame(pred = factor(), actual = factor())
  # Use K-fold cross validation
  K = 5
  for (k in 1:K) {
    # shuffle the data
    res <- get_train_test(0.8, data.sub)
    fit.knn <-
      knn(
        train = res$X.train,
        test = res$X.test,
        cl = res$Y.train
      )
    classifications <-
      rbind(classifications,
            data.frame(pred = fit.knn, actual = res$Y.test))
  }
  confusionMatrix(classifications$pred, classifications$actual)
}
```

Decision Trees Model

Similar to the kNN, let's make a function which allows us the subset the entire dataset to contain songs only of a particular genre which will give us the accuracy of our algorithm using the decision trees model. Why we subset our data to include only specific genres will become clear later.

```
dtree_function <- function(data, genres) {
  data.sub <- data[data$Genre %in% genres,]
  data.sub$Genre <- droplevels(data.sub$Genre)
  res <- get_train_test(0.8, data.sub)

  # Decision Tree
  set.seed(103)
  fit.dtree <-
    train(
      Genre ~ .,
      data = res$data.train,
      method = "rpart",
      parms = list(split = "information")
    )

  Y.pred.dtree <-
    predict(fit.dtree, newdata = data.frame(res$X.test), type = "raw")
  confusionMatrix(Y.pred.dtree, res$Y.test)
}
```

PILOT RESULTS

Let's sample 10 genres at random from the dataset and see what our accuracy is

```r
set.seed(3)
genres <- sample(levels(datax$Genre), 10)
knn_function(datax, genres)$overall["Accuracy"]
```

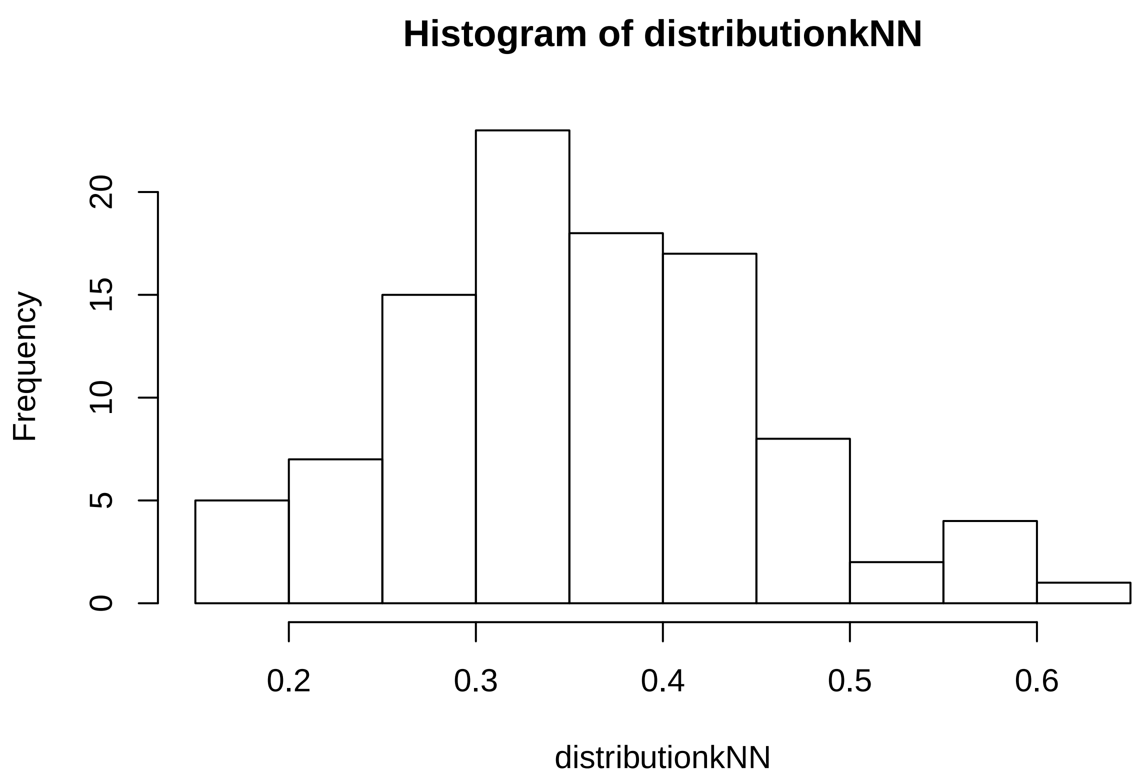
```
##  Accuracy
## 0.2753036
```

```r
dtree_function(datax, genres)$overall["Accuracy"]
```

```
##  Accuracy
## 0.3076923
```

Here's the histogram of the frequency distribution of the accuracy

```r
set.seed(1)
distributionkNN <- replicate(100,{
  genres <- sample(levels(datax$Genre), 10)
  knn_function(datax, genres)$overall["Accuracy"]
})

hist(distributionkNN)
```



As we can see, a vast majority of the accuracy distribution is less than 0.5. Let's select 10 genres that we know to be sufficiently different from each other in terms of their genome composition and see if that increases the accuracy much.

```r
genres <-
  list(
    "canadianpop",
    "electronica",
    "rock",
    "modernblues",
    "r&b",
    "polishblackmetal",
    "videogamemusic",
    "irishfolk",
    "koreanpop",
    "hiphop"
  )
knn_function(datax, genres)$overall["Accuracy"]
```

```
##  Accuracy
## 0.4301639
```

```r
dtree_function(datax, genres)$overall["Accuracy"]
```

```
## Accuracy
##      0.5
```

ANALYSIS

As we can see, the accuracy is incredibly poor, even when we picked genres that seem to be different. Let's try to analyze the cause of this by analysing.

kNN Model

```
genres <-
  list(
    "canadianpop",
    "electronica",
    "rock",
    "modernblues",
    "r&b",
    "polishblackmetal",
    "videogamemusic",
    "irishfolk",
    "koreanpop",
    "hiphop"
  )
knn_function(datax, genres)
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction         canadianpop electronica hiphop irishfolk koreanpop
##   canadianpop              355          24    105        36         7
##   electronica               15          36      7         2         2
##   hiphop                   103           8    526        41         3
##   irishfolk                 24           2     28         7         0
##   koreanpop                  3           3      5         6         0
##   modernblues              140          31     99        27         2
##   polishblackmetal           0           1      0         0         0
##   r&b                        9           3     18        12         1
##   rock                      21          11     18        12         2
##   videogamemusic            37           3     49        14         2
##                   Reference
## Prediction         modernblues polishblackmetal r&b rock videogamemusic
##   canadianpop              152                0  22   24             40
##   electronica               16                0   2    4              3
##   hiphop                   130                1  17   20             52
##   irishfolk                 35                0   9    4             12
##   koreanpop                  3                0   0    1              0
##   modernblues              311                5  28   32             50
##   polishblackmetal           2                0   1    0              1
##   r&b                       25                1  10    0              0
##   rock                      43                0   7   36              3
##   videogamemusic            47                0   4    6             31
##
## Overall Statistics
##
##                Accuracy : 0.4302
##                  95% CI : (0.4125, 0.448)
##     No Information Rate : 0.2803
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2781
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: canadianpop Class: electronica Class: hiphop
## Sensitivity                      0.5021            0.29508        0.6152
## Specificity                      0.8250            0.98258        0.8292
## Pos Pred Value                   0.4641            0.41379        0.5838
## Neg Pred Value                   0.8460            0.97098        0.8469
## Prevalence                       0.2318            0.04000        0.2803
## Detection Rate                   0.1164            0.01180        0.1725
## Detection Prevalence             0.2508            0.02852        0.2954
## Balanced Accuracy                0.6636            0.63883        0.7222
##                      Class: irishfolk Class: koreanpop Class: modernblues
## Sensitivity                  0.044586         0.000000             0.4071
## Specificity                  0.960595         0.993072             0.8189
## Pos Pred Value               0.057851         0.000000             0.4290
## Neg Pred Value               0.948788         0.993727             0.8052
## Prevalence                   0.051475         0.006230             0.2505
## Detection Rate               0.002295         0.000000             0.1020
## Detection Prevalence         0.039672         0.006885             0.2377
## Balanced Accuracy            0.502590         0.496536             0.6130
##                      Class: polishblackmetal Class: r&b Class: rock
## Sensitivity                         0.000000   0.100000     0.28346
## Specificity                         0.998357   0.976610     0.95997
## Pos Pred Value                      0.000000   0.126582     0.23529
## Neg Pred Value                      0.997701   0.969707     0.96859
## Prevalence                          0.002295   0.032787     0.04164
## Detection Rate                      0.000000   0.003279     0.01180
## Detection Prevalence                0.001639   0.025902     0.05016
## Balanced Accuracy                   0.499178   0.538305     0.62172
##                      Class: videogamemusic
## Sensitivity                        0.16146
## Specificity                        0.94332
## Pos Pred Value                     0.16062
## Neg Pred Value                     0.94365
## Prevalence                         0.06295
## Detection Rate                     0.01016
## Detection Prevalence               0.06328
## Balanced Accuracy                  0.55239
```

As we can see from the results, several genres are more obscure, and/or have only a small number of songs in the given dataset. Consequently, the KNN algorithm will find very few neighbors of these genres when trying to classify any given point. Therefore, it makes sense that the classification accuracy and other stats would be poor, since KNN makes a decision based on label popularity. If we only have 5 nearby labels (genres) to look at, and each one is different (due to the low proportion of songs in each of the nearby genres), then it's essentially a toss-up for assigning a predicted label.

Let's analyse the **Decision Tree** model:

```r
genres <-
  list(
    "canadianpop",
    "electronica",
    "rock",
    "modernblues",
    "r&b",
    "polishblackmetal",
    "videogamemusic",
    "irishfolk",
    "koreanpop",
    "hiphop"
  )
dtree_function(datax, genres)
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction         canadianpop electronica hiphop irishfolk koreanpop
##   canadianpop              101           7     31        10         2
##   electronica                0           0      0         0         0
##   hiphop                    16           2    111         2         0
##   irishfolk                  3           0      0         3         0
##   koreanpop                  0           0      0         0         0
##   modernblues               44           1     15        11         1
##   polishblackmetal           0           0      0         0         0
##   r&b                        0           0      0         0         0
##   rock                       0           0      0         0         0
##   videogamemusic             3          12     17         2         0
##                   Reference
## Prediction         modernblues polishblackmetal r&b rock videogamemusic
##   canadianpop               54                0   6    9              0
##   electronica                0                0   0    0              0
##   hiphop                    13                0   6    4              1
##   irishfolk                  3                0   0    0              3
##   koreanpop                  0                0   0    0              0
##   modernblues               64                2   3   16              0
##   polishblackmetal           0                0   0    0              0
##   r&b                        0                0   0    0              0
##   rock                       0                0   0    0              0
##   videogamemusic             6                0   0    0             26
##
## Overall Statistics
##
##                Accuracy : 0.5
##                  95% CI : (0.4596, 0.5404)
##     No Information Rate : 0.2852
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3451
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: canadianpop Class: electronica Class: hiphop
## Sensitivity                      0.6048            0.00000        0.6379
## Specificity                      0.7314            1.00000        0.8991
## Pos Pred Value                   0.4591                NaN        0.7161
## Neg Pred Value                   0.8308            0.96393        0.8615
## Prevalence                       0.2738            0.03607        0.2852
## Detection Rate                   0.1656            0.00000        0.1820
## Detection Prevalence             0.3607            0.00000        0.2541
## Balanced Accuracy                0.6681            0.50000        0.7685
##                      Class: irishfolk Class: koreanpop Class: modernblues
## Sensitivity                   0.107143         0.000000             0.4571
## Specificity                   0.984536         1.000000             0.8021
## Pos Pred Value                0.250000              NaN             0.4076
## Neg Pred Value                0.958194         0.995082             0.8322
## Prevalence                    0.045902         0.004918             0.2295
## Detection Rate                0.004918         0.000000             0.1049
## Detection Prevalence          0.019672         0.000000             0.2574
## Balanced Accuracy             0.545839         0.500000             0.6296
##                      Class: polishblackmetal Class: r&b Class: rock
## Sensitivity                         0.000000    0.00000     0.00000
## Specificity                         1.000000    1.00000     1.00000
## Pos Pred Value                           NaN        NaN         NaN
## Neg Pred Value                      0.996721    0.97541     0.95246
## Prevalence                          0.003279    0.02459     0.04754
## Detection Rate                      0.000000    0.00000     0.00000
## Detection Prevalence                0.000000    0.00000     0.00000
## Balanced Accuracy                   0.500000    0.50000     0.50000
##                      Class: videogamemusic
## Sensitivity                        0.86667
## Specificity                        0.93103
## Pos Pred Value                     0.39394
## Neg Pred Value                     0.99265
## Prevalence                         0.04918
## Detection Rate                     0.04262
## Detection Prevalence               0.10820
## Balanced Accuracy                  0.89885
```

Decision tree classifiers are great for both binary and multi-class problems. They make decisions based on the values (branches) of attributes (leaves / nodes) of the thing being classified, traversing a tree-like structure until reaching a final classification. So in our case, the leaves would be song attributes like "tempo" or "danceability", and the branches would be the different values each attribute can take. As we saw before, having very few songs in a particular genre is the most likely culprit of our poor results. There's also the issue of class imbalance, where some genres (like Hip Hop and Canadian pop) have a far greater percentage of songs than other genres do.

TRYING TO IMPROVE ACCURACY

Let's see what the most popular genres are.

```
descending <- datax %>% group_by(Genre) %>% summarise(n = n()) %>% arrange(desc(n))
descending
```

```
## # A tibble: 626 x 2
##    Genre                    n
##    <fct>                <int>
##  1 alternativeamericana  1891
##  2 electrolatino         1009
##  3 doo-wop                972
##  4 reading                969
##  5 nuelectro              909
##  6 groovemetal            903
##  7 psychill               901
##  8 deepdeephouse          892
##  9 torontoindie           884
## 10 newrave                875
## # ... with 616 more rows
```

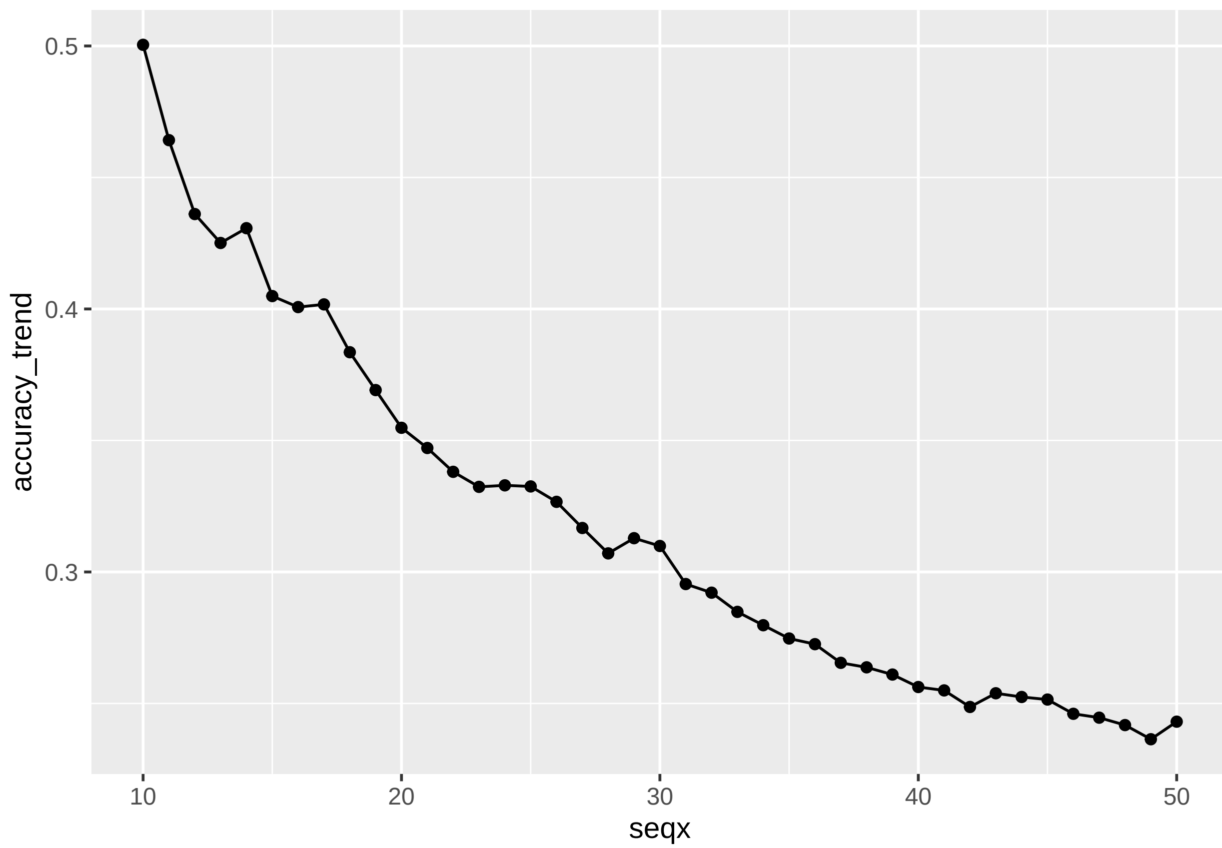
If we reduce the genres to only those that are very common, the kNN Model in theory should show a considerable improvement as any given label will have plenty of neighbours. Let's try by keeping only the top 30 most popular genres.

```
genre_top30 <- descending$Genre[1:30]
new_data <- filter(datax, Genre %in% genre_top30)

knn_function(new_data, genre_top30)$overall["Accuracy"]
```

```
##  Accuracy
## 0.3098714
```

This accuracy is actually worse than when we hand-picked the genres. Let's see the trend in this accuracy as we pick the top 50 to the top 10 most common genres.

```
seqx <- 50:10
top_trend <- function(x){
  genre_topx <- descending$Genre[1:x]
  new_datax <- filter(datax, Genre %in% genre_topx)
  knn_function(new_datax, genre_topx)$overall["Accuracy"]
}

accuracy_trend <- sapply(seqx, top_trend)
ggplot(data.frame(accuracy_trend), aes(x = seqx, y = accuracy_trend)) + geom_point() + geom_line()
```



Even in our best case scenario, when we use only 10 genres, our accuracy is still only 0.500441

CONCLUSION

No matter what we try and even when we consider the practically impossible scenario of only including the top 10 most common genres, are accuracy does not increase much beyond 0.5. This means that there is massive overlap within the genres. The possible cause for this could be that spotify divides their data into these many genres to aid their machine learning algorithm which predicts user behaviour and not genre. Let's see some overlapping genres:

```r
genre_grouped <- datax %>% group_by(Genre) %>%
  summarise(dance = mean(Danceability),
            energy = mean(Energy),
            key = mean(Key),
            loudness = mean(Loudness),
            mode = mean(Mode),
            speechness = mean(Speechness),
            acousticness = mean(Acousticness),
            intstrumentalness = mean(Instrumentalness),
            liveness = mean(Liveness),
            valence = mean(Valence),
            tempo = mean(Tempo),
            duration = mean(Duration_ms),
            Time = mean(time_signature)) %>%
  arrange(Genre)

genre_grouped
```

```
## # A tibble: 626 x 14
##    Genre dance energy   key loudness   mode speechness acousticness
##    <fct> <dbl>  <dbl> <dbl>    <dbl>  <dbl>      <dbl>        <dbl>
##  1 ""    0.606  4.42  -7.42    0.577 0.0765     0.463       0.00234
##  2 "abs~ 0.600  0.627  5.31  -10.5   0.5        0.0835      0.247
##  3 "aca~ 0.554  0.578  5.86   -6.88  0.630      0.0807      0.398
##  4 "aco~ 0.540  0.557  4.61   -6.69  0.794      0.0372      0.406
##  5 "afr~ 0.495  0.695  5.19   -6.62  0.893      0.0864      0.356
##  6 "afr~ 0.616  0.668  5.66   -9.00  0.588      0.0660      0.298
##  7 "ala~ 0.521  0.575  4.84   -8.46  0.772      0.0459      0.352
##  8 "alb~ 0.736  0.709  5.70   -6.39  0.394      0.184       0.159
##  9 "alb~ 0.565  0.723  5.51   -5.53  0.919      0.0407      0.190
## 10 "alb~ 0.489  0.621  5.71   -8.74  0.636      0.0556      0.258
## # ... with 616 more rows, and 6 more variables: intstrumentalness <dbl>,
## #   liveness <dbl>, valence <dbl>, tempo <dbl>, duration <dbl>, Time <dbl>
```

Let's see the overlap between Swedish Death Metal and Polish Black Metal:

```
print(genre_grouped[c(which(genre_grouped$Genre == "swedishdeathmetal"),
                      which(genre_grouped$Genre == "polishblackmetal")), ], width = Inf)
```

```
## # A tibble: 2 x 14
##   Genre             dance energy   key loudness  mode speechness acousticness
##   <fct>             <dbl>  <dbl> <dbl>    <dbl> <dbl>      <dbl>        <dbl>
## 1 swedishdeathmetal 0.242  0.935  5.18    -6.62 0.702     0.109     0.00107
## 2 polishblackmetal  0.220  0.868  4.3     -6.32 0.6       0.0985    0.0000567
##   intstrumentalness liveness valence  tempo duration  Time
##               <dbl>    <dbl>   <dbl>  <dbl>    <dbl> <dbl>
## 1             0.595    0.226   0.221 22908.   256855.  3.95
## 2             0.774    0.227   0.157 26842.   382170   3.4
```

What about Emo Pop and Indonesian Punk Pop

```
print(genre_grouped[c(which(genre_grouped$Genre == "indonesianpoppunk"),
                      which(genre_grouped$Genre == "popemo")), ], width = Inf)
```

```
## # A tibble: 2 x 14
##   Genre             dance energy   key loudness  mode speechness acousticness
##   <fct>             <dbl>  <dbl> <dbl>    <dbl> <dbl>      <dbl>        <dbl>
## 1 indonesianpoppunk 0.472  0.853  4.96    -4.78 0.888     0.0647       0.0647
## 2 popemo            0.478  0.863  4.88    -4.52 0.734     0.0661       0.0288
##   intstrumentalness liveness valence  tempo duration  Time
##               <dbl>    <dbl>   <dbl>  <dbl>    <dbl> <dbl>
## 1            0.0221    0.200   0.505 26980.   224151.  3.97
## 2            0.0236    0.214   0.512 26510.   206123.  3.91
```

Both of the genres in these sits have striking similarities and these are just a few of the 626 genres that Spotify uses to classify its music data.

Having these varied genres definitely aids in predicting user behaviour, similar to the Netflix Movie Prediction Program we did in the last course. But having these many genres also means that there is going to be major overlap and the distinction between genres is going to be very less. Our current genomes don't have enough range to appropriately house this variety of genres and still produce a desirable result.

What works as a boon when it comes to predicting user behaviour works as a bane when it comes to predicting genres, thus serving as the main limitation.

Perhaps in the future, if the data we recieved had the name of the artist, our algorithm would perform much better because artists don't usually tend to stray too far away from their genre (other than our occasional Bob Dylan). Also there could be a genre classification system that strikes a balance between user prediction and genre prediction.