KNN Implementation for predicting song popularity using Spotify Dataset

Nisha Selvarajan

26 Sept 2020

Contents

Objectives:

Why Do Some Songs Become Popular? DJ Khaled boldly claimed to always know when a song will be a hit. We decided to further investigate by asking three key questions: Are there certain characteristics for hit songs, what are the largest influencers on a song's success, and can old songs even predict the popularity of new songs? Predicting how popular a song will be is no easy task. To answer these questions, we made use of the spotify Song Dataset, and use knn machine learning to predict. We will finally present a model that can predict how likely a song will be a hit, with more than 85% accuracy.

Data Description

- 27K Rows, with 14 columns. You can download data on the link https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks

```r
library(knitr)
library(kableExtra)

df <- data.frame(Names = c("acousticness","danceability","energy",
                          "duration_ms","instrumentalness","valence",
                          "popularity","liveness","loudness",
                          "speechiness","year","mode",
                          "key","artists","genre"),
                 Description = c("Numerical-confidence measure from 0.0
                 to 1.0 of whether the track is acoustic. 1.0 represents
                 high confidence the track is acoustic.","Numerical-
                 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.","Numerical-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. For example, death metal has high energy,
```

```
                 while a Bach prelude scores low on the scale." ,"Numerical-
                 The duration of the track in milliseconds."," Numerical-
                 Detects the presence of an audience in the recording.
                 Higher liveness values represent an increased probability
                 that the track was performed live. A value above 0.8 provides
                 strong likelihood that the track is live.","","Numerical-
                 A measure from 0.0 to 1.0 describing the musical positiveness
                 conveyed by a track. Tracks with high valence sound more
                 positive (e.g. happy, cheerful, euphoric), while tracks
                 with low valence sound more negative (e.g. sad, depressed,
                 angry).", "Numerical-The higher the value the more popular
                 the song is.Ranges from 0 to 1","Numerical-The overall
                 loudness of a track in decibels (dB). Loudness values are
                 averaged across the entire track and are useful for
                 comparing relative loudness of tracks. Loudness is the
                 quality of a sound that is the primary psychological correlate
                 of physical strength (amplitude). Values typical range
                 between -60 and 0 db.", " Numerical-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. Values above 0.66
                 describe tracks that are probably made entirely of spoken words.
                 Values between 0.33 and 0.66 describe tracks that may contain
                 both music and speech, either in sections or layered, including
                 such cases as rap music.Values below 0.33 most likely represent
                 music and other non-speech-like tracks.","Numerical-Ranges from
                 1921 to 2020","Categorical-(0 = Minor, 1 = Major)","Categorical-
                 All keys on octave encoded as values ranging from 0 to 11,
                 starting on C as 0, C# as 1 and so on…","Categorical-List of artists
                 mentioned","Categorical-genre of the song"))


kbl(df) %>%
  kable_paper(full_width = F) %>%
  column_spec(2, width = "30em")
```

Names	Description
acousticness	Numerical-confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
danceability	Numerical- 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	Numerical-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. For example, death metal has high energy, while a Bach prelude scores low on the scale.
duration_ms	Numerical- The duration of the track in milliseconds.
instrumentalness	Numerical- Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
valence	
popularity	Numerical- A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
liveness	Numerical-The higher the value the more popular the song is.Ranges from 0 to 1
loudness	Numerical-The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.
speechiness	Numerical-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. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music.Values below 0.33 most likely represent music and other non-speech-like tracks.
year	Numerical-Ranges from 1921 to 2020
mode	Categorical-(0 = Minor, 1 = Major)
key	Categorical- All keys on octave encoded as values ranging from 0 to 11, starting on C as 0, C# as 1 and so on…
artists	Categorical-List of artists mentioned
genre	Categorical-genre of the song

Using k-nearest neighbours to predict the song popularity

- ***Step 1: import dataset***

```
library(class)
library(gmodels)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
spotify=read.csv(file = "/Users/nselvarajan/spotifydata/spotify.csv", sep = ",")
spotify <- data.frame(spotify, stringsAsFactors = FALSE)
head(spotify)
```

```
##                                                              artists
## 1                                  "Cats" 1981 Original London Cast
## 2                                         "Cats" 1983 Broadway Cast
## 3                       "Fiddler On The Roof" Motion Picture Chorus
## 4                    "Fiddler On The Roof" Motion Picture Orchestra
## 5   "Joseph And The Amazing Technicolor Dreamcoat" 1991 London Cast
## 6 "Joseph And The Amazing Technicolor Dreamcoat" 1992 Canadian Cast
##   acousticness danceability duration_ms    energy instrumentalness  liveness
## 1    0.5750833    0.4427500    247260.0 0.3863358      0.022717397 0.2877083
## 2    0.8625385    0.4417308    287280.0 0.4068077      0.081158264 0.3152154
## 3    0.8565714    0.3482857    328920.0 0.2865714      0.024592949 0.3257857
## 4    0.8849259    0.4250741    262891.0 0.2457704      0.073587279 0.2754815
## 5    0.6054444    0.4373333    232428.1 0.4293333      0.037533560 0.2161111
## 6    0.6095556    0.4872778    205091.9 0.3099056      0.004695657 0.2747667
##    loudness speechiness     tempo   valence popularity key mode count
## 1 -14.20542  0.18067500 115.98350 0.3344333   38.00000   5    1    12
## 2 -10.69000  0.17621154 103.04415 0.2688654   33.07692   5    1    26
## 3 -15.23071  0.11851429  77.37586 0.3548571   34.28571   0    1     7
## 4 -15.63937  0.12320000  88.66763 0.3720296   34.44444   0    1    27
## 5 -11.44722  0.08600000 120.32967 0.4586667   42.55556  11    1     9
## 6 -18.26639  0.09802222 118.64894 0.4415556   34.16667   5    1    36
##         genres
## 1 'show tunes'
## 2  'dance pop'
## 3  'dance pop'
## 4  'dance pop'
## 5  'dance pop'
## 6  'dance pop'
```

- ***Step 2: Clean the Data***

```r
spotify<-subset(spotify,select = c(acousticness ,danceability
                                   ,energy,instrumentalness,liveness,
                                   loudness,speechiness,tempo,valence,
                                   popularity))

spotify$popularity[spotify$popularity>0.5]  <- 'Y'
spotify$popularity[spotify$popularity==0.5]  <- 'N/Y'
spotify$popularity[spotify$popularity<0.5]  <- 'N'
spotify$popularity<-as.factor(spotify$popularity)
str(spotify)
```

```
## 'data.frame':    27621 obs. of  10 variables:
##  $ acousticness    : num  0.575 0.863 0.857 0.885 0.605 ...
##  $ danceability    : num  0.443 0.442 0.348 0.425 0.437 ...
##  $ energy          : num  0.386 0.407 0.287 0.246 0.429 ...
##  $ instrumentalness: num  0.0227 0.0812 0.0246 0.0736 0.0375 ...
##  $ liveness        : num  0.288 0.315 0.326 0.275 0.216 ...
##  $ loudness        : num  -14.2 -10.7 -15.2 -15.6 -11.4 ...
##  $ speechiness     : num  0.181 0.176 0.119 0.123 0.086 ...
##  $ tempo           : num  116 103 77.4 88.7 120.3 ...
```

```
##  $ valence         : num  0.334 0.269 0.355 0.372 0.459 ...
##  $ popularity      : Factor w/ 3 levels "N","N/Y","Y": 3 3 3 3 3 3 1 3 3 3 ...
```

- ***Step 3: Data Splicing***
  - The kNN algorithm is applied to the training data set and the results are verified on the test data set.
  - I used 25% to test data and 75% to data train.
  - After obtaining training and testing data sets, then we will create a separate data frame which has values to be compared with actual final values

```
indxTrain <- createDataPartition(y = spotify$popularity,p = .75,list = FALSE)
training <- spotify[indxTrain,]
testing <- spotify[-indxTrain,]
```

- ***Step 4:Data Pre-Processing With Caret***
  - The scale transform calculates the standard deviation for an attribute and divides each value by that standard deviation.
  - The center transform calculates the mean for an attribute and subtracts it from each value.
  - Combining the scale and center transforms will standardize your data.
  - Attributes will have a mean value of 0 and a standard deviation of 1.
  - The caret package in R provides a number of useful data transforms.
  - Training transforms can prepared and applied automatically during model evaluation.
  - Transforms applied during training are prepared using the preProcess() and passed to the train() function via the preProcess argument.

```
trainX <- training[,names(training) != "popularity"]
preProcValues <- preProcess(x = trainX,method = c("center", "scale"))
```

- ***Step 5:Model Training and Tuning***
  - To control parameters for train, a trainControl function is used.
  - The option "repeatedcv" method controls the number of repetitions for resampling used in repeated K-fold cross-validation.

```
set.seed(400)
ctrl <- trainControl(method="repeatedcv",repeats = 3)
```

Performance improvement techniques and improved accuracy achieved.

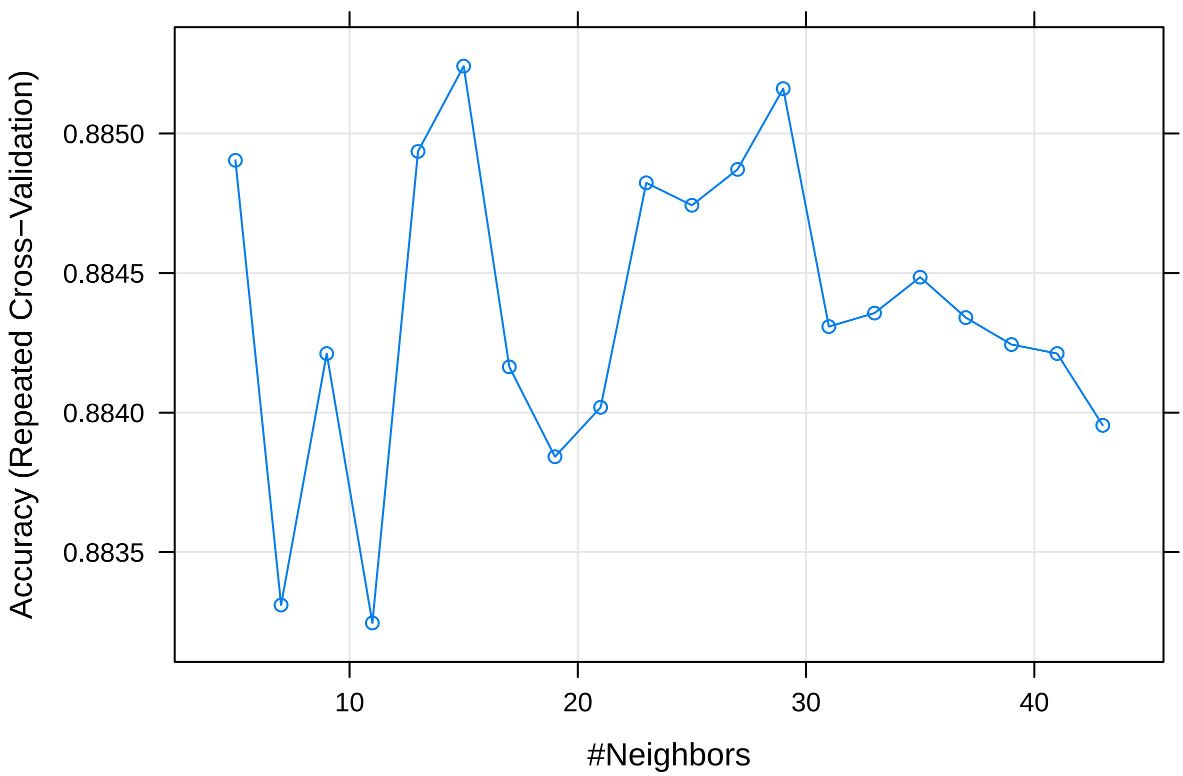
- ***Step 6:How to choose value for K to improve performance***
- Time to fit a knn model using caret with preprocessed values.
- From the output of the model,maximum accuracy(0.8842111) is achieved by k = 27.
- We can observe accuracy for different types of k.

```
knnFit <- train( popularity~ ., data = training, method = "knn",
                 trControl = ctrl, preProcess = c("center","scale"),
                 tuneLength = 20)
knnFit
```

```
## k-Nearest Neighbors
##
## 20716 samples
##     9 predictor
##     3 classes: 'N', 'N/Y', 'Y'
##
## Pre-processing: centered (9), scaled (9)
```

```
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 18645, 18644, 18645, 18644, 18644, 18645, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8849039  0.5282902
##    7  0.8833104  0.5169813
##    9  0.8842115  0.5146932
##   11  0.8832463  0.5054532
##   13  0.8849359  0.5085339
##   15  0.8852415  0.5055794
##   17  0.8841638  0.4979614
##   19  0.8838419  0.4941368
##   21  0.8840190  0.4918714
##   23  0.8848234  0.4935559
##   25  0.8847427  0.4900539
##   27  0.8848715  0.4894960
##   29  0.8851610  0.4893743
##   31  0.8843082  0.4847336
##   33  0.8843566  0.4836602
##   35  0.8844853  0.4822096
##   37  0.8843403  0.4806894
##   39  0.8842439  0.4792240
##   41  0.8842117  0.4768426
##   43  0.8839543  0.4736060
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 15.
```

```r
plot(knnFit)
```



- ***Step 7: Making predictions***
- We build knn by using training & test data sets. After building the model, then we can check the accuracy of forecasting using confusion matrix.

```
knnPredict <- predict(knnFit,newdata = testing )
```

Interpretation of the results and prediction accuracy achieved

- ***Evaluate the model performance***
- The accuracy of our model on the testing set is 88%.
- We can visualise the model's performance using a confusion matrix.
- We can evaluvate the accuracy, precision and recall on the training and validation sets to evaluate the performance of knn algorithm.

```
confusionMatrix(knnPredict, testing$popularity )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N  N/Y    Y
##        N    521    4  227
##        N/Y    0    0    0
##        Y    563   10 5580
##
## Overall Statistics
##
##                Accuracy : 0.8836
##                  95% CI : (0.8758, 0.891)
##     No Information Rate : 0.841
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5014
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: N Class: N/Y Class: Y
## Sensitivity           0.48063   0.000000   0.9609
## Specificity           0.96032   1.000000   0.4781
## Pos Pred Value        0.69282        NaN   0.9069
## Neg Pred Value        0.90850   0.997972   0.6981
## Prevalence            0.15699   0.002028   0.8410
## Detection Rate        0.07545   0.000000   0.8081
## Detection Prevalence  0.10891   0.000000   0.8911
## Balanced Accuracy     0.72047   0.500000   0.7195
```

```
mean(knnPredict == testing$popularity)
```

```
## [1] 0.8835626
```

Overall insights obtained from the implemented project

- Overall accuracy of the model is 88%.It is safe to assume that knn models can be trained on the audio feature data to predict the popularity.
- Sensitivity for popular song is 0.52030 and for unpopular song is 0.9571.

- Specificity for popular song is 0.95619 and for unpopular song is 0.5200