Predicting Music Genres

Samuel Kuenti

12 5 2022

Abstract

The aim of this project was to predict the music genre of a song based on a number of features of the song. In order to achieve this, a knn-model was compared to a random forest approach. It was expected that while the random forest model would perform reasonably well, the knn-model would be overwhelmed with the number of predictors used. This was confirmed by testing and comparing the models on actual data. Both models could not be optimised fully due to limitations imposed by long computation times.

Table 1: Variables in the dataset

x	x	x	x	x	x	x
instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms

x	x	x	x	x	x	x	x
energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo

x	x	x
obtained_date	valence	music_genre

Introduction

The aim of this project is to develop a machine learning algorithm to predict the genre of a song based on a number of attributes. Such an algorithm could support the automatic organisation of a collection of digital music.

The project guidelines ask for at least two different modeling approaches to be compared side by side. As will be discussed below, the data in this case provides multiple predictors. Based on theory, simpler modelling approaches like k-nearest neighbors should suffer when too many predictors are involved. In order to explore this theoretical caveat, a k-nearest-neighbor model will be compared to a random forest approach, which should deal better with a larger number of inputs.

Analysis

Data Preparation

Obtaining Data This project is based on data compiled into a CSV file by Gaoyuan. The dataset is available from the public kaggle data repository. Due to issues with implementing kaggle's authentication process in R, the data file was copied to this project's GitHub repository and is loaded from there.

```
## [1] "2022-07-20 23:32:53 CEST"
```

Initial data inspection

The following table shows what kind of information the dataset contains.

Data cleaning An initial look at the raw data reveals that some songs seem to have incomplete information (e.g. tempo = ?, or a duration of -1ms).

instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	obtained_date	valence	music_genre
32894	Röyksopp	Röyksopp's Night Out	27	0.00468	0.652	-1	0.941	0.79200	A#	0.115	-5.201	Minor	0.0748	100.889	4-Apr	0.759	Electronic
46652	Thievery Corporation	The Shining Path	31	0.0127	0.622	218293	0.890	0.95000	D	0.124	-7.043	Minor	0.0300	115.00200000000001	4-Apr	0.531	Electronic
30097	Dillon Francis	Hurricane	28	0.00306	0.620	215613	0.755	0.01180	G#	0.534	-4.617	Major	0.0345	127.994	4-Apr	0.333	Electronic

instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	obtained_date	valence	music_genre
62177	Dubloadz	Nitro	34	0.0254	0.774	166875	0.700	0.00253	C#	0.157	-4.498	Major	0.2390	128.014	4-Apr	0.270	Electronic
24907	What So Not	Divide & Conquer	32	0.00465	0.638	222369	0.587	0.90900	F#	0.157	-6.266	Major	0.0413	145.036	4-Apr	0.323	Electronic
89064	Axel Boman	Hello	47	0.00523	0.755	519468	0.731	0.85400	D	0.216	-10.517	Minor	0.0412	?	4-Apr	0.614	Electronic

```
## 'data.frame':    50005 obs. of  18 variables:
##  $ instance_id     : num  32894 46652 30097 62177 24907 ...
##  $ artist_name     : chr  "Röyksopp" "Thievery Corporation" "Dillon Francis" "Dubloadz" ...
##  $ track_name      : chr  "Röyksopp's Night Out" "The Shining Path" "Hurricane" "Nitro" ...
##  $ popularity      : num  27 31 28 34 32 47 46 43 39 22 ...
##  $ acousticness    : num  0.00468 0.0127 0.00306 0.0254 0.00465 ...
##  $ danceability    : num  0.652 0.622 0.62 0.774 0.638 0.755 0.572 0.809 0.509 0.578 ...
##  $ duration_ms     : num  -1 218293 215613 166875 222369 ...
##  $ energy          : num  0.941 0.89 0.755 0.7 0.587 0.731 0.803 0.706 0.921 0.731 ...
##  $ instrumentalness: num  7.92e-01 9.50e-01 1.18e-02 2.53e-03 9.09e-01 8.54e-01 7.74e-06 9.03e-01 2.
##  $ key             : chr  "A#" "D" "G#" "C#" ...
##  $ liveness        : num  0.115 0.124 0.534 0.157 0.157 0.216 0.106 0.0635 0.178 0.111 ...
##  $ loudness        : num  -5.2 -7.04 -4.62 -4.5 -6.27 ...
##  $ mode            : chr  "Minor" "Minor" "Major" "Major" ...
##  $ speechiness     : num  0.0748 0.03 0.0345 0.239 0.0413 0.0412 0.351 0.0484 0.268 0.173 ...
##  $ tempo           : chr  "100.889" "115.00200000000001" "127.994" "128.014" ...
##  $ obtained_date   : chr  "4-Apr" "4-Apr" "4-Apr" "4-Apr" ...
##  $ valence         : num  0.759 0.531 0.333 0.27 0.323 0.614 0.23 0.761 0.273 0.203 ...
##  $ music_genre     : chr  "Electronic" "Electronic" "Electronic" "Electronic" ...
```

|| || || ||

Also, some variables have unsuitable data types (e.g. tempo is). This will be corrected. In addition, negative durations of songs will be replaced with NAs (these will be suppressed later).

```
## Warning: NAs durch Umwandlung erzeugt
```

As noted above, there were some problematic table entries, which lead to the generation of NAs.

Table 3: Columns with NAs

	x
instance_id	TRUE
artist_name	FALSE
track_name	FALSE
popularity	TRUE
acousticness	TRUE
danceability	TRUE
duration_ms	TRUE
energy	TRUE
instrumentalness	TRUE

	x
key	FALSE
liveness	TRUE
loudness	TRUE
mode	FALSE
speechiness	TRUE
tempo	TRUE
obtained_date	FALSE
valence	TRUE
music_genre	FALSE

There are different options for dealing with NA rows. Here, for simplicity's sake and because the dataset is reasonably large, the lamest option will be implemented - all songs with NAs will be removed from the dataset.

All in all, there are data on 50005 songs in the dataset (including those with NAs). Those with NAs will be deleted.

This leads to the loss of about 20% of the data and leaves complete information about 40560 songs, more than enough as will be seen later on.

Table 4: Columns with NAs after data cleaning

	x
instance_id	FALSE
artist_name	FALSE
track_name	FALSE
popularity	FALSE
acousticness	FALSE
danceability	FALSE
duration_ms	FALSE
energy	FALSE
instrumentalness	FALSE
key	FALSE
liveness	FALSE
loudness	FALSE
mode	FALSE
speechiness	FALSE
tempo	FALSE
obtained_date	FALSE
valence	FALSE
music_genre	FALSE

Now the data looks as follows:

```
## 'data.frame':    40560 obs. of  18 variables:
##  $ instance_id     : num  46652 30097 62177 24907 43760 ...
##  $ artist_name     : Factor w/ 6864 levels "","!!!","? & The Mysterians",..: 6204 1590 1735 6626 306
##  $ track_name      : chr  "The Shining Path" "Hurricane" "Nitro" "Divide & Conquer" ...
##  $ popularity      : num  31 28 34 32 46 43 39 22 30 27 ...
##  $ acousticness    : num  0.0127 0.00306 0.0254 0.00465 0.0289 0.0297 0.00299 0.00934 0.855 0.0337 .
##  $ danceability    : num  0.622 0.62 0.774 0.638 0.572 0.809 0.509 0.578 0.607 0.513 ...
##  $ duration_ms     : num  218293 215613 166875 222369 214408 ...
##  $ energy          : num  0.89 0.755 0.7 0.587 0.803 0.706 0.921 0.731 0.158 0.828 ...
##  $ instrumentalness: num  9.50e-01 1.18e-02 2.53e-03 9.09e-01 7.74e-06 9.03e-01 2.76e-04 1.12e-02 0.
##  $ key             : Factor w/ 13 levels "","A","A#","B",..: 7 13 6 11 4 12 10 2 11 4 ...
##  $ liveness        : num  0.124 0.534 0.157 0.157 0.106 0.0635 0.178 0.111 0.106 0.109 ...
##  $ loudness        : num  -7.04 -4.62 -4.5 -6.27 -4.29 ...
##  $ mode            : Factor w/ 3 levels "","Major","Minor": 3 2 2 2 2 3 3 3 3 3 ...
##  $ speechiness     : num  0.03 0.0345 0.239 0.0413 0.351 0.0484 0.268 0.173 0.0345 0.0609 ...
##  $ tempo           : num  115 128 128 145 150 ...
##  $ obtained_date   : chr  "4-Apr" "4-Apr" "4-Apr" "4-Apr" ...
##  $ valence         : num  0.531 0.333 0.27 0.323 0.23 0.761 0.273 0.203 0.307 0.0591 ...
##  $ music_genre     : Factor w/ 11 levels "","Alternative",..: 7 7 7 7 7 7 7 7 7 7 ...
##  - attr(*, "na.action")= 'omit' Named int [1:9445] 1 6 14 17 25 33 36 37 40 41 ...
##   ..- attr(*, "names")= chr [1:9445] "1" "6" "14" "17" ...
```

Light mode

In order to limit the processing times, only 50% of the actual data are used.

(Note: with 5-fold crossvalidation and 50% of the data, the script took about half an hour to compute on a 2017 iMac.)

Also, crossvalidation will be n-fold, with n being set at 5 initially. Later on this may be increased to 10, based on processing speeds.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## Warning in createDataPartition(music_raw_data$music_genre, times = 1, p =
## percent/100, : Some classes have no records ( ) and these will be ignored
```

Caution: even with these limits, the whole script will take a long time to run through (it took about 20 minutes on a 2017 iMac).

Splitting of data

The data will be split into a training set (80% of the data), and a test set (20%). The training set will be used to tune the models. The test set will simulate actual new data in order to assess the final performance of the various models.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## Warning in createDataPartition(music_raw_data$music_genre, times = 1, p = 0.2, :
## Some classes have no records ( ) and these will be ignored
```

Modelling

The idea is to predict the genre of a song (music_genre). The genre variable distinguishes the following musical genres:

Table 5: Musical Genres

x
Electronic
Anime
Jazz
Alternative
Country
Rap
Blues
Rock
Classical
Hip-Hop

```
## [1] 10
```

Some of the parameters in the dataset seem pretty quantifiable, for instance based on the envelope of a song (e.g. key, tempo, popularity), while others are subjective and would have to be rated by listeners (e.g. danceability, energy, valence). A viable initial approach could be to only include objectively quantifiable predictors, in order to produce a lean and sleek algorithm. Subjective parameters based on listeners' ratings would only be included if initial results prove to be too inaccurate.

The following variables are assumed to be based on listeners' subjective rating (there is no information on how the dataset was obtained):

- acousticness
- danceability (maybe correlated with tempo?)
- energy
- instrumentalness
- liveness
- speechiness
- valence

This leaves the following basic predictors, which are assumed to be objectively measurable by suitable algorithms, without involving actual listeners (with popularity being a bit of a hybrid, it will be included here):

- popularity
- key
- loudness
- mode
- tempo
- duration

The artist name would be a subjective parameter with high predictive power (many artists tend to be in some genre for most of their career). However, being a factor with several thousand levels, this feature was found to increase computation times beyond practical limits, so it is excluded from the models.

The parameters above look fairly generic, and it is doubtful that a song's genre could be reliably predicted based on these parameters alone. Nevertheless, this will be attempted in a first step, and 'subjective parameters' (those based on listeners' ratings) will be included later to see whether they contribute to model performance.

The following variables are deemed irrelevant for the current task and will not be included in the models:

- ID-number of the song (instance_id)
- track name (track_name)
- obtained date (obtained_date)

The models will predict the song's genre (music_genre).

Models with six objective parameters

As described above, a first attempt will only involve parameters which can potentially be derived from the music by means of appropriate algorithms (e.g. based on the envelope curve). It will be interesting to compare the performance of the two models with a different number of predictors.

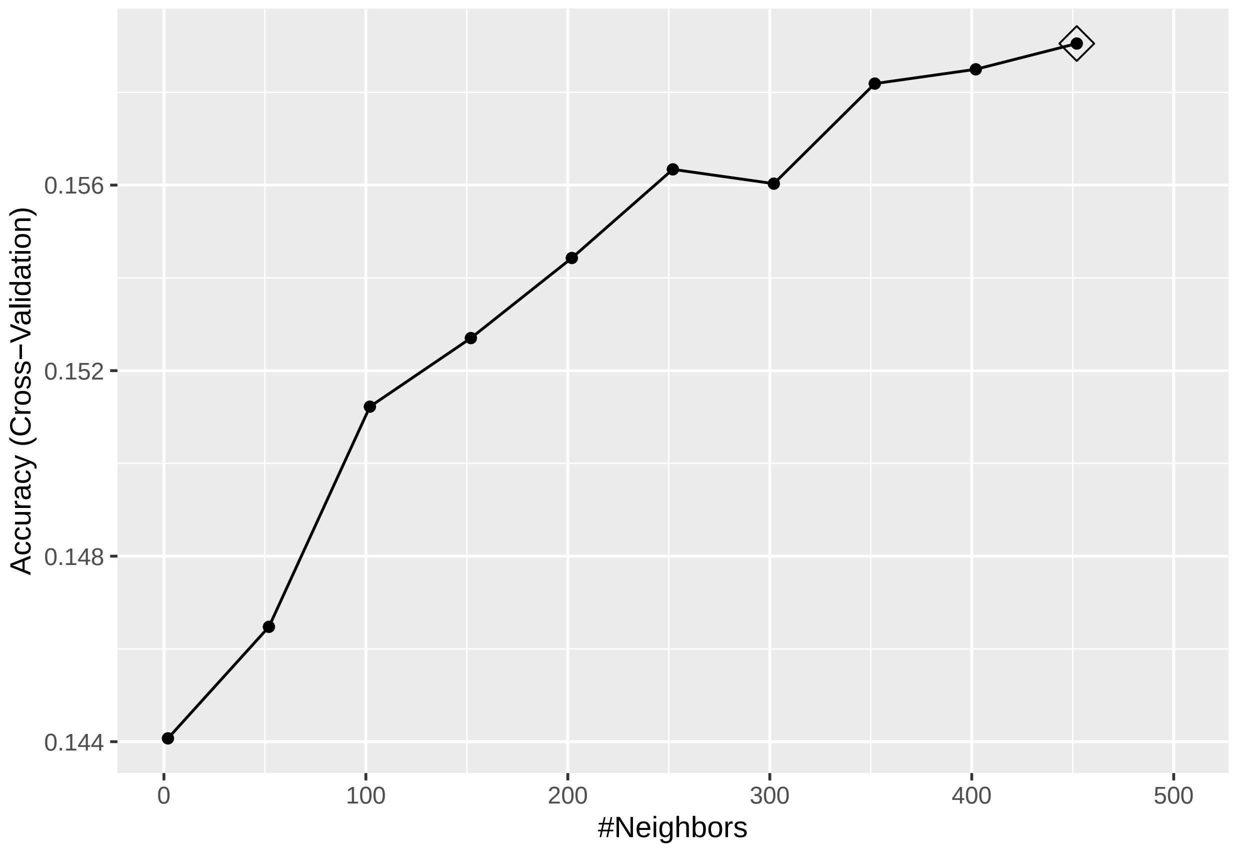
As discussed above, the following predictors are used as 'objective parameters':

- popularity
- key
- loudness
- mode
- tempo
- duration

knn (six predictors) A knn-model will be fitted using the six objective predictors above. In order to somewhat limit computation times, 5-fold cross-validation will be used to speed up the model fitting, and initial resolution will be low.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used

## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used

## Warning: predictions failed for Fold1: k=502 Error in knn3Train(train = structure(c(28, 32, 43, 22,
##   too many ties in knn

## Warning: predictions failed for Fold2: k=502 Error in knn3Train(train = structure(c(28, 43, 22, 27,
##   too many ties in knn

## Warning: predictions failed for Fold3: k=502 Error in knn3Train(train = structure(c(28, 32, 43, 22,
##   too many ties in knn

## Warning: predictions failed for Fold4: k=502 Error in knn3Train(train = structure(c(32, 43, 31, 50,
##   too many ties in knn

## Warning: predictions failed for Fold5: k=502 Error in knn3Train(train = structure(c(28, 32, 22, 27,
##   too many ties in knn

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.

## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```



Note that while the model's accuracy seems to increase with larger ks, at k = 502, the training function reports 'too many ties in knn'. As more neighbors are included into the voting process, the likelihood of ties occurring seems to increase, and in this case, the limit of viable ks seems to be around 500.

In a second step, k-values approaching this limit at k=502 (the first observed k raising an error above) will be examined with better resolution.

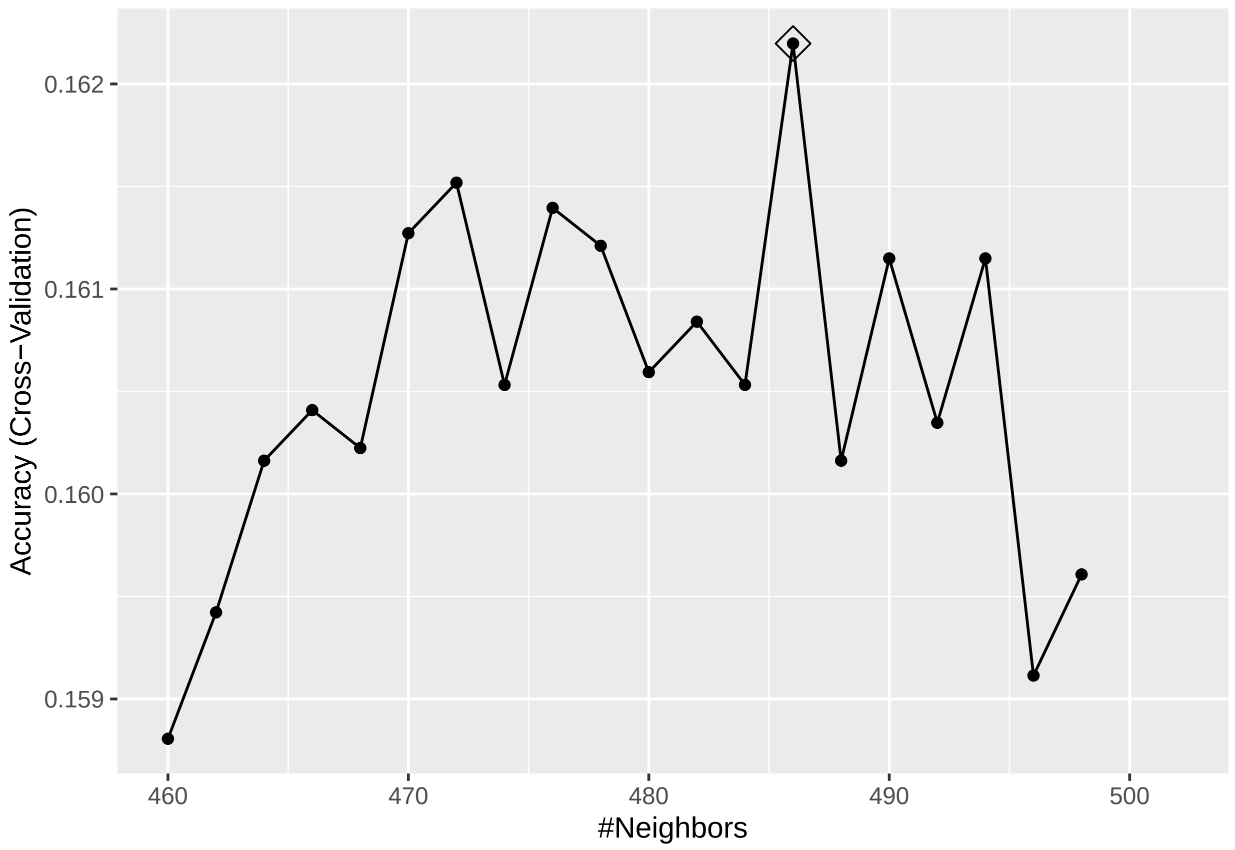
```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```

```
## Warning: predictions failed for Fold1: k=500 Error in knn3Train(train = structure(c(28, 32, 43, 22, 
##   too many ties in knn
```

```
## Warning: predictions failed for Fold1: k=502 Error in knn3Train(train = structure(c(28, 32, 43, 22, 
##   too many ties in knn
```

```
## Warning: predictions failed for Fold2: k=500 Error in knn3Train(train = structure(c(28, 43, 22, 27, 
##   too many ties in knn
## Warning: predictions failed for Fold2: k=502 Error in knn3Train(train = structure(c(28, 43, 22, 27, 3
##   too many ties in knn

## Warning: predictions failed for Fold3: k=500 Error in knn3Train(train = structure(c(28, 32, 43, 22, 2
##   too many ties in knn

## Warning: predictions failed for Fold3: k=502 Error in knn3Train(train = structure(c(28, 32, 43, 22, 2
##   too many ties in knn

## Warning: predictions failed for Fold4: k=500 Error in knn3Train(train = structure(c(32, 43, 31, 50, 2
##   too many ties in knn

## Warning: predictions failed for Fold4: k=502 Error in knn3Train(train = structure(c(32, 43, 31, 50, 2
##   too many ties in knn

## Warning: predictions failed for Fold5: k=500 Error in knn3Train(train = structure(c(28, 32, 22, 27, 3
##   too many ties in knn

## Warning: predictions failed for Fold5: k=502 Error in knn3Train(train = structure(c(28, 32, 22, 27, 3
##   too many ties in knn

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.

## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results

## Warning: Removed 2 rows containing missing values (geom_point).

## Warning: Removed 2 row(s) containing missing values (geom_path).
```



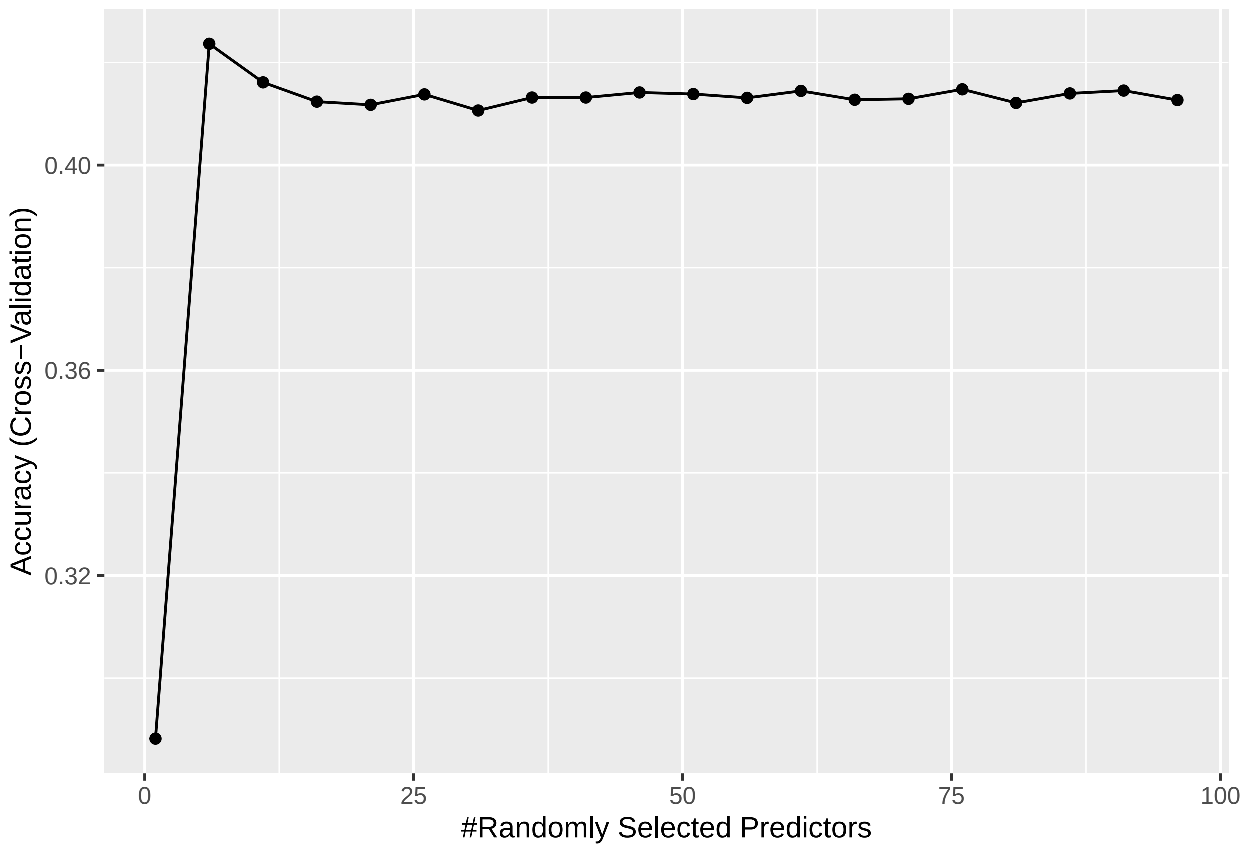
```
## [1] 486
```

Random forest (seven predictors) For comparison, a random forest model will be trained on the train_set.

Fitting random forests weighs heavy on the CPU, so the minimum nodesize is set to 5 (instead of the default of 1) in order to limit computation time somewhat.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```



```
## [1] 6
```

This model did best when 6 variables were selected for each tree, with accuracy declining slightly as more were selected.

Models with six objective in addition to seven subjective parametes

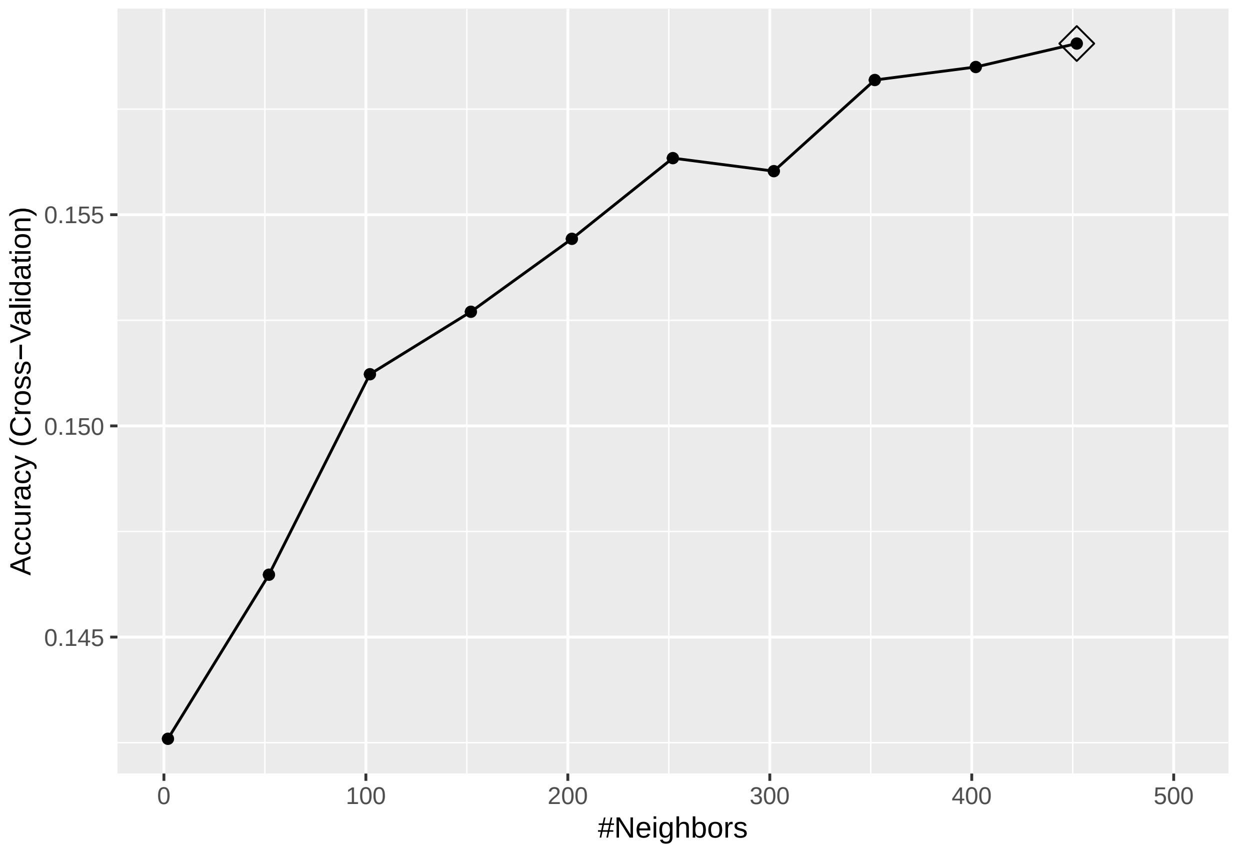
Here, seven additional 'subjective' parameters are included into the model:

- acousticness
- danceability (my favorite predictor!)
- energy
- instrumentalness
- liveness
- speechiness
- valence

The idea is to find out whether including these is worth the trouble, since assessing these parameters would inevitably involve collecting user ratings. This would be much more complicated than just deriving the 'subjective six' parameters directly from the music by means of suitable algorithms.

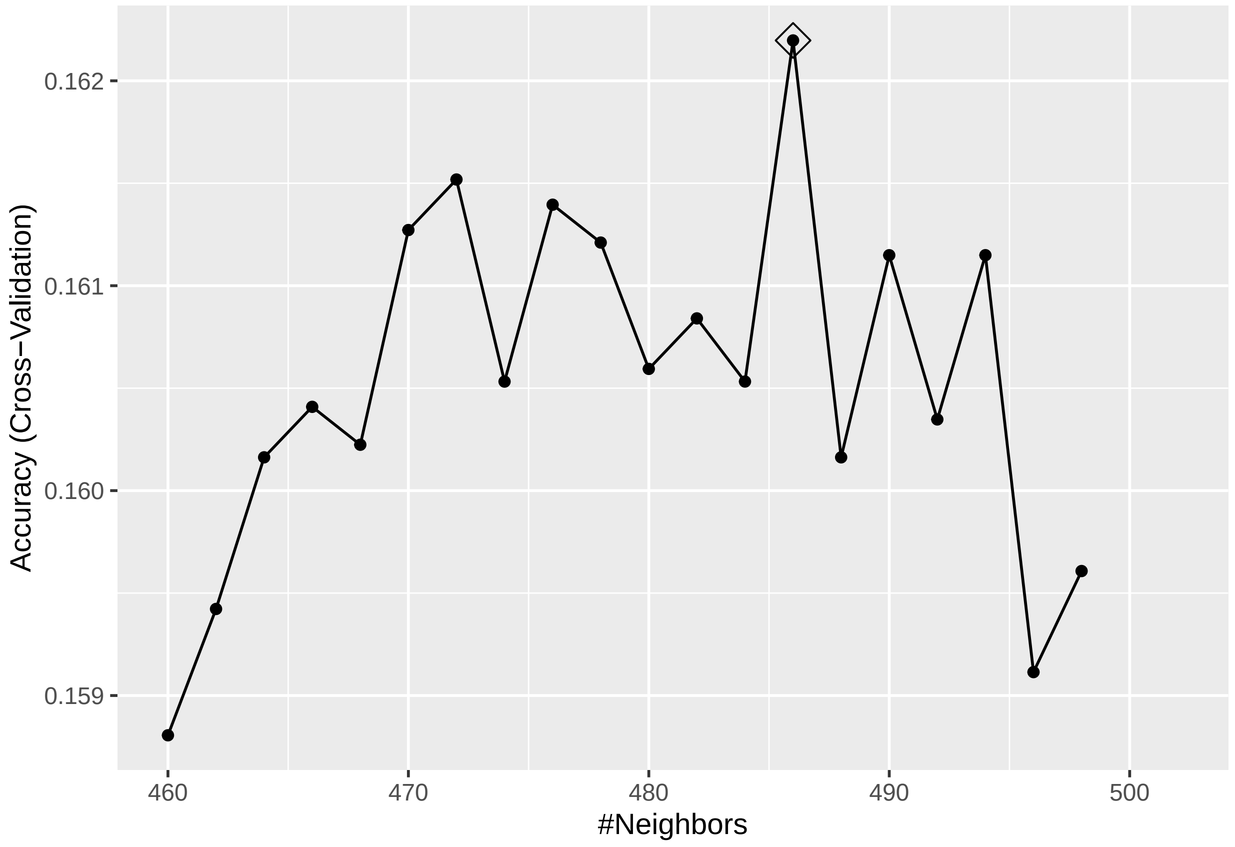
knn (13 predictors) This new knn-model will be fitted using the five objective predictors plus the seven additional ones above. Since the knn-model involving the 'objective six' only performed best at high values of k, k-values over a large range will be tried with low resolution in a first step. This should reveal whether high values of k are optimal again.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used

## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used

## Warning: predictions failed for Fold1: k=502 Error in knn3Train(train = structure(c(28, 32, 43, 22,
##   too many ties in knn

## Warning: predictions failed for Fold2: k=502 Error in knn3Train(train = structure(c(28, 43, 22, 27,
##   too many ties in knn

## Warning: predictions failed for Fold3: k=502 Error in knn3Train(train = structure(c(28, 32, 43, 22,
##   too many ties in knn

## Warning: predictions failed for Fold4: k=502 Error in knn3Train(train = structure(c(32, 43, 31, 50,
##   too many ties in knn

## Warning: predictions failed for Fold5: k=502 Error in knn3Train(train = structure(c(28, 32, 22, 27,
##   too many ties in knn

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.

## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results

## Warning: Removed 1 rows containing missing values (geom_point).

## Warning: Removed 1 row(s) containing missing values (geom_path).
```



Indeed, a high number of neigbors improves performance, but the limit leading to 'too many ties' is again reached at k around 500. So in a second step, ks approaching 500 are tested with better resolution.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used

## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used

## Warning: predictions failed for Fold1: k=500 Error in knn3Train(train = structure(c(28, 32, 43, 22, 
##   too many ties in knn

## Warning: predictions failed for Fold1: k=502 Error in knn3Train(train = structure(c(28, 32, 43, 22, 
##   too many ties in knn

## Warning: predictions failed for Fold2: k=500 Error in knn3Train(train = structure(c(28, 43, 22, 27, 
##   too many ties in knn

## Warning: predictions failed for Fold2: k=502 Error in knn3Train(train = structure(c(28, 43, 22, 27, 
##   too many ties in knn

## Warning: predictions failed for Fold3: k=500 Error in knn3Train(train = structure(c(28, 32, 43, 22, 
##   too many ties in knn

## Warning: predictions failed for Fold3: k=502 Error in knn3Train(train = structure(c(28, 32, 43, 22, 
##   too many ties in knn
## Warning: predictions failed for Fold4: k=500 Error in knn3Train(train = structure(c(32, 43, 31, 50, 2
##   too many ties in knn

## Warning: predictions failed for Fold4: k=502 Error in knn3Train(train = structure(c(32, 43, 31, 50, 2
##   too many ties in knn

## Warning: predictions failed for Fold5: k=500 Error in knn3Train(train = structure(c(28, 32, 22, 27, 3
##   too many ties in knn

## Warning: predictions failed for Fold5: k=502 Error in knn3Train(train = structure(c(28, 32, 22, 27, 3
##   too many ties in knn

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.

## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results

## Warning: Removed 2 rows containing missing values (geom_point).

## Warning: Removed 2 row(s) containing missing values (geom_path).
```



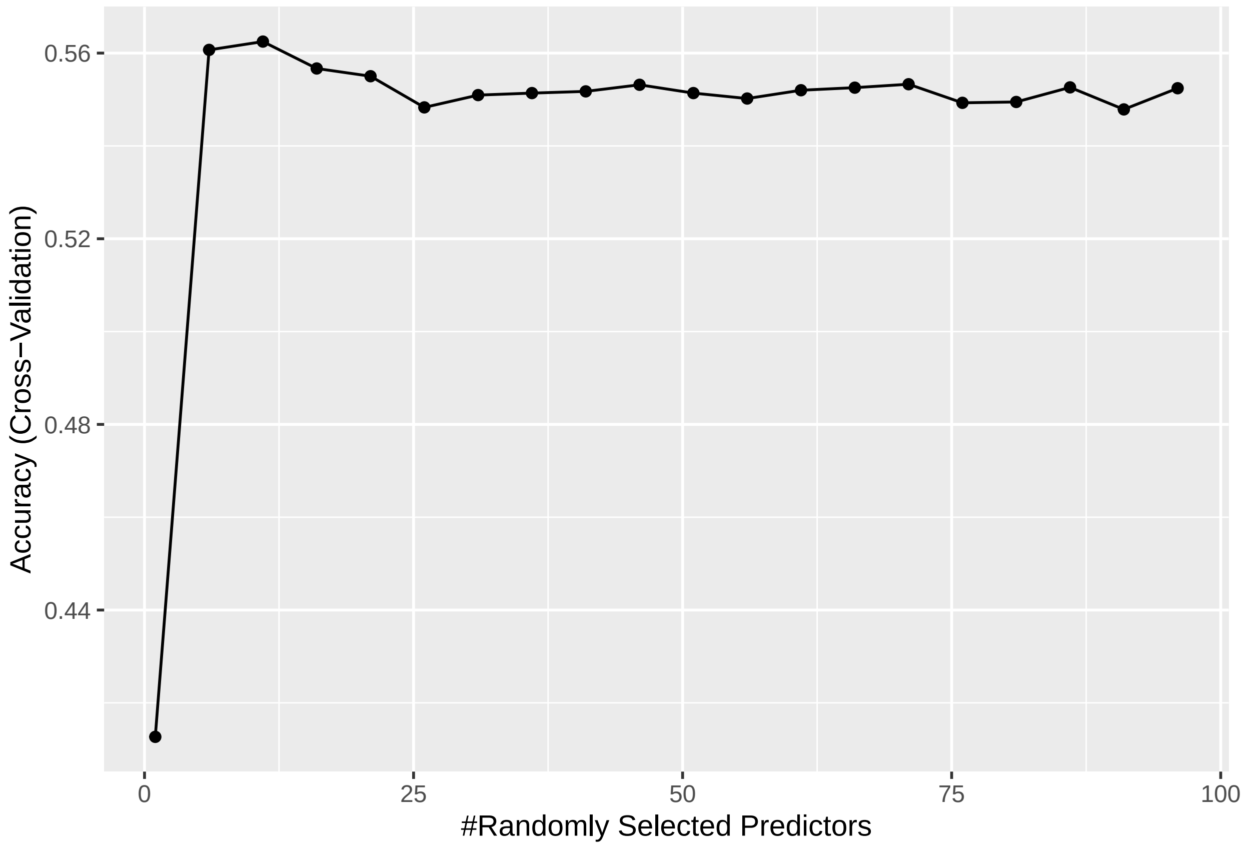
```
## [1] 486
```

The accuracy of this second model on the test set looks as follows:

Random forest (12 predictors) Finally, a random forest model with all predictors will be trained on the train_set.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```



```
## [1] 11
```

Again, best performance is obtained with few predictors, it dips with more being added.

Results

The performance of the knn model was the same, whether 6 or 13 predictors were included. Both models performed best with a k just below a threshold of 500, where the modelling function raised an error because 'too many ties' affected the voting process. The best k for the six-predictors knn model was 486, the best k for the full knn-model was 486.

The random forest model with six predictors performed best when 6 variables were sampled for each tree, and the full model worked best with 11 variables.

The following tabe summarises the overal accuracies of the four models on the test set:

Table 6: Model accuracies

Method	Accuracy
knn (6 predictors)	0.16326
Random forest (6 predictors)	0.42132
knn (12 predictors)	0.16326
Random forest (12 predictors)	0.42009

```
## [1] "2022-07-21 00:04:16 CEST"
```

Note that both knn-models reached the same accuracy, no matter how many predictors were included. Also, they performed barely above chance levels. Interestingly, the random forest model performed better with fewer predictors.

Conclusion

The aim of this project was to compare two different modeling approaches to predict music genres based on a fairly large dataset with information about songs. The random forest approache was expected to work reasonably well, while the knn model was predicted to suffer from the 'curse of dimensionality' with the number of predictors included in the models.

These were the main observations:

- classification based on knn performs terrible with that many predictors, with model performance barely above chance levels
- normalising predictors could have helped the knn model somewhat, although it is doubtful whether the effort would have had a significant effect on model performance
- for some reason, the 'too many ties issue' arises at k = 500, I can't mathematically tell why (the limit exist independent of the size of the dataset used; I worked with 10% of the data for most of the development process to limit processing times, but the limit was the same when I used the larger proportions of the entire dataset)
- the knn-model did not improve when more predictors were added; in fact, it looked and performed exactly the same, no mather whether six or thirteen predictors were included
- the knn approach may already be suffering from the 'curse of dimensionality' in the leaner version with seven predictors, as the predictor space is complex and distances tend to be large, while differences in distances remaining relatively subtle; an issue which is only exacerbated by adding more predictors
- model fitting requires very long computation times if all data is included, this imposes limit to cross-validation and optimisation parameters
- as expected, the ramdom forest model performed reasonably well
- however, the random forest model performed better with fewer predictors, which was against expectations

While the knn-approach seems to be unsuitable for this problem, a solution based on an optimised random forest model could be a viable path to solving the automatic classification of songs in a music library. Additional effort would be required to optimse the model further, as several modelling parameters were kept constant here in order to allow for practical computation times.