MovieLens EDX Capstone Project Report

Author: MM

9/29/2021

Contents

Summary

This is a report for the capstone project required to obtain the EDX Data Science Professional Certification issued by HarvardX.

The aim was to create a movie recommendation system using a subset of the MovieLens data set, limited to 10 million records.

I have divided the resulting data set in two portions:

- edx - I used this as my training set
- validation - I used this as my test set

I documented the project through the following files:

- An Rmd file report detailing the methodology applied,
- A PDF report (issued from the Rmd file above mentioned), and
- An R script file that generates the predicted movie ratings and calculates the root-mean-square error (RMSE).

Results

The following table shows the final results obtained with the various models:

```
rmse_results %>% knitr::kable()
```

method	RMSE	approx_accuracy
Guessing_benchmark1	1.9417280	0.0997791
Simple_Average_benchmark2	1.0612018	0.5080073
Regularised_Lineal_Model	0.8258487	0.6171213

Please note that the approx_accuracy is a derived calculation from the Guessing_benchmark1 to illustrate the meaning of change on RMSE. It does not represent a direct comparison of the actual and predicted ratings.

Methodology

Libraries

The following libraries are required for proper execution of the R script file:

```
library(plyr)
library(tidyverse)
library(caret)
library(data.table)
library(stringr)
library(knitr)
library(purrr)
library(skimr)
library(randomForest)
library(corrplot)
library(rpart)
library(brnn)
library(monomvn)
```

Creating the dataset

The following code was provided by EDX as part of the project requirements and introduction. I applied this code to generate the data set used in the project.

```
############################################################
# Create edx set, validation set (final hold-out test set)
############################################################

# MovieLens 10M data set:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
```

```
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Data Exploration

I performed the following data exploration steps:

- Understanding the training and test datasets:

```
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  17 variables:
##  $ userId              : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId             : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating              : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp           : int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 8
##  $ title               : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)
##  $ genres              : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-
Fi|Thriller" "Action|Adventure|Sci-Fi" ...
##  $ release_year        : num  1992 1995 1995 1994 1994 ...
##  $ userId_stand        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ movieId_stand       : num  0.00186 0.00283 0.00447 0.00484 0.00504 ...
##  $ timestamp_stand     : num  0.112 0.112 0.112 0.112 0.112 ...
```

```
##  $ genres_encoded_stand: num  0 0.00126 0.00251 0.00377 0.00503 ...
##  $ title_encoded_stand : num  0.00 9.37e-05 1.87e-04 2.81e-04 3.75e-04 ...
##  $ release_year_stand  : num  0.828 0.86 0.86 0.849 0.849 ...
##  $ u_effect            : num  -1.49 -1.49 -1.49 -1.49 -1.49 ...
##  $ m_effect            : num  0.57 0.406 0.122 0.197 0.21 ...
##  $ ry_effect           : num  -6.48e-17 2.81e-17 2.81e-17 2.18e-16 2.18e-16 ...
##  $ gr_effect           : num  5.33e-17 -2.86e-16 -1.31e-16 2.28e-17 -2.24e-16 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
str(validation)
```

```
## Classes 'data.table' and 'data.frame':   999999 obs. of  8 variables:
##  $ userId       : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId      : num  231 480 586 151 858 ...
##  $ rating       : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp    : int  838983392 838983653 838984068 868246450 868245645 868245920 1136075494 113357
##  $ title        : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy (19
##  $ genres       : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Drama|Ro
##  $ movieId_stand: num  0.00353 0.00735 0.00898 0.0023 0.01316 ...
##  $ userId_stand : num  0.0 0.0 0.0 1.4e-05 1.4e-05 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

*Obtaining descriptive statistics for each column in the training data set

```
skimmed <- skim(edx)
```

```
skimmed[, c(1:4, 8:17)]
```

Table 2: Data summary

Name	edx
Number of rows	9000055
Number of columns	6
Key	NULL
Column type frequency:	
character	2
numeric	4
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	n_unique	whitespace
title	0	1	10676	0
genres	0	1	797	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
userId	0	1	3.586982e+04	2.058525e+04	1.0	18124	35738	53607	71567	
movieId	0	1	4.121700e+03	8.902411e+03	1.0	648	1834	3626	65133	
rating	0	1	3.510000e+00	1.060000e+00	0.5	3	4	4	5	
timestamp	0	1	1.032616e+09	1.409667e+07	7.896520e+08	9.467683e+08	1.035493e+09	1.126750e+09	1.231131e+09	

- Ratings distribution

```
edx %>%
  group_by(rating) %>%
  summarise (count = n() ) %>%
  rename("Rating"=rating, "Frequency"=count) %>%
  mutate("%"=paste(round(100*Frequency/nrow(edx),2),"%",sep="")) %>%
  knitr::kable( align = "lcr")
```

Rating	Frequency	%
0.5	85374	0.95%
1.0	345679	3.84%
1.5	106426	1.18%
2.0	711422	7.9%
2.5	333010	3.7%
3.0	2121240	23.57%
3.5	791624	8.8%
4.0	2588430	28.76%
4.5	526736	5.85%
5.0	1390114	15.45%

- Movies with more ratings

```
edx %>%
  group_by(title) %>%
  summarise (count =n() ) %>%
  ungroup() %>% top_n(.,10) %>%
  rename("# of ratings"=count) %>%
  knitr::kable()
```

```
## Selecting by count
```

title	# of ratings
Apollo 13 (1995)	24284
Braveheart (1995)	26212
Forrest Gump (1994)	31079
Fugitive, The (1993)	25998
Jurassic Park (1993)	29360
Pulp Fiction (1994)	31362
Shawshank Redemption, The (1994)	28015

title	# of ratings
Silence of the Lambs, The (1991)	30382
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25672
Terminator 2: Judgment Day (1991)	25984

- Evidence that there are movies with low number of ratings

```
edx %>%
  group_by(title) %>%
  summarise (count =n() ) %>%
  ungroup() %>% top_n(.,-10) %>%
  tail( .,10)  %>%
    rename("# of ratings"=count) %>%
  knitr::kable()
```

```
## Selecting by count
```

title	# of ratings
Twice Upon a Time (1983)	1
Uncle Nino (2003)	1
Valerie and Her Week of Wonders (Valerie a tÃ½den divu) (1970)	1
Variety Lights (Luci del varietÃ) (1950)	1
Vinci (2004)	1
When Time Ran Out… (a.k.a. The Day the World Ended) (1980)	1
Where A Good Man Goes (Joi gin a long) (1999)	1
Won't Anybody Listen? (2000)	1
Young Unknowns, The (2000)	1
Zona Zamfirova (2002)	1

- Top active users

```
edx %>%
  group_by(userId) %>%
  summarise (count =n() ) %>%
  ungroup() %>% top_n(.,10) %>%
  rename("# of ratings"=count) %>%
  knitr::kable(align = "lr")
```

```
## Selecting by count
```

userId	# of ratings
3817	3733
14463	4648
19635	3771
27468	4023
27584	3142
58357	3361

userId	# of ratings
59269	6616
63134	3371
67385	6360
68259	4036

- Less active users

```
edx %>%
  group_by(userId) %>%
  summarise (count =n() ) %>%
  ungroup() %>% top_n(.,-10) %>%
  tail( .,10)  %>%
  rename("# of ratings"=count) %>%
  knitr::kable(align = "lr")
```

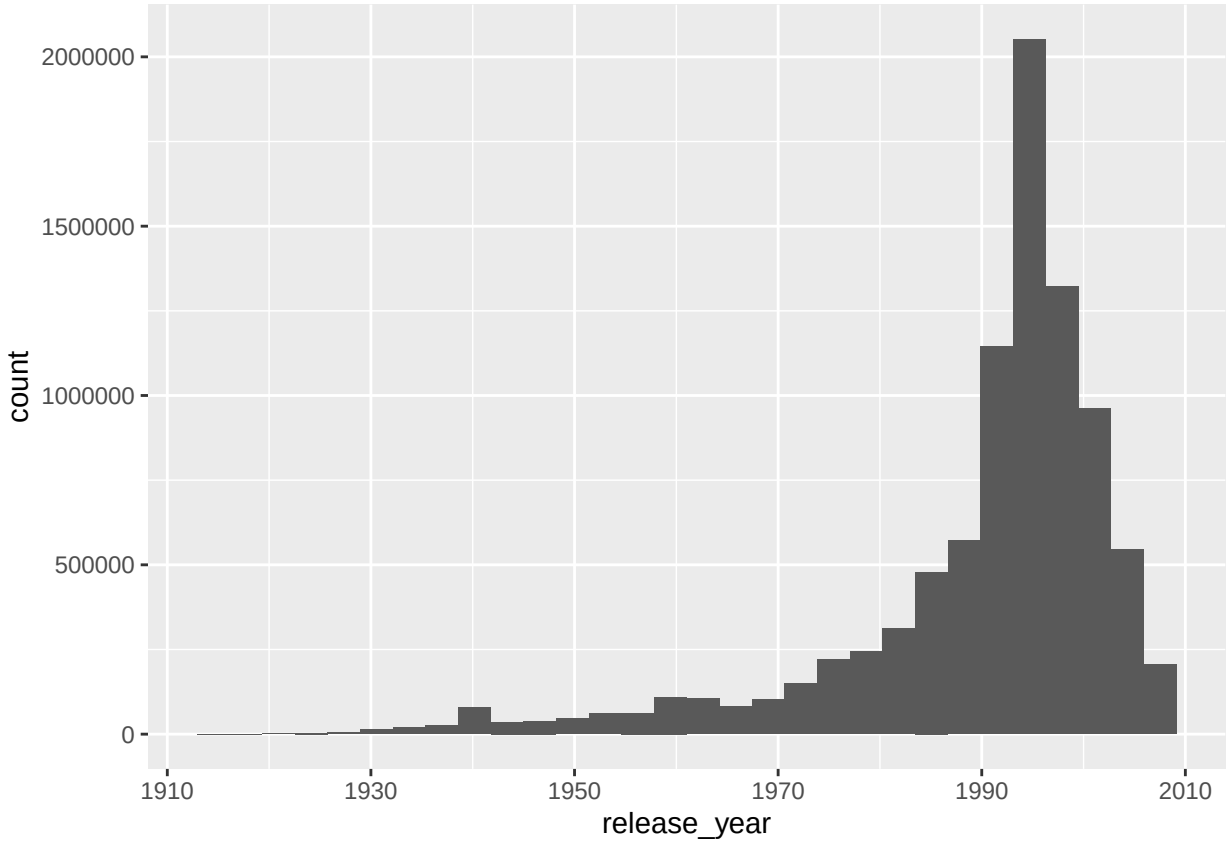
```
## Selecting by count
```

userId	# of ratings
49034	14
50608	13
52674	14
57894	14
62317	14
62516	10
63143	14
68161	14
68293	14
71344	14

- Identifying release year to be use as possible predictor

```
edx <- edx %>% mutate(release_year = as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/)]$"), rege

edx %>% ggplot() + geom_histogram(aes(release_year))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



- ratings values available

```
unique(edx$rating)
```

```
##  [1] 5.0 3.0 2.0 4.0 4.5 3.5 1.0 1.5 2.5 0.5
```

Benchmarking: two methods

As starting point I created two basic models that serve as benchmark:

- Guessing the account: this method uses a categorical approach towards the rating and allows to establish a baseline for overall accuracy to aid interpretability of the results. The benchmark would approximate the proportion of categories available in the rating as the probability of guessing the outcome.
- Naive model: this method provide a more realistic benchmark for the RMSE and is based on assigning the overall average rating to all the predictions.

Also a table to collect the results of the various models and allow comparison was created:

```
#defining a function for RMSE

RMSE <- function(true_ratings, predicted_ratings){
```

```
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

# guessing the outcome - benchmark1
y_hat_guess <- sample(c(seq(0.5,5, by=0.5)), length(validation$rating), replace = TRUE)
acc_guessing <-mean(y_hat_guess == validation$rating)
guess_rmse <- RMSE(validation$rating, y_hat_guess)

#naive model:  average rate for all - benchmark 2

mu_hat <- mean(edx$rating)
naive_rmse <- RMSE(validation$rating, mu_hat)
acc_naive <- 1 - ((naive_rmse*(1-acc_guessing))/guess_rmse)

#creating table that will store all results

rmse_results <- data_frame(method = "Guessing_benchmark1",
                           RMSE = guess_rmse,
                           approx_accuracy= acc_guessing)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Simple_Average_benchmark2",
                                     RMSE = naive_rmse, approx_accuracy= acc_naive ))
```

Preprocessing: Encoding character variables

I used the Helmert method to encode character variables into numerical vectors for use in the model. Helmert encoding is a form of contrast encoding where each value of a categorical variable is compared to the mean of the subsequent levels. I have added a column to the output of the encode_helmert function below to obtain unique encoded items for each value of the variables "genres" and "title", and then merged it back to the training data set.

```
#defining the encoding function
helmert <- function(n) {
  m <- t((diag(seq(n-1, 0)) - upper.tri(matrix(1, n, n)))[-n,])
  t(apply(m, 1, rev))
}
encode_helmert <- function(df, var) {
  x <- df[[var]]
  x <- unique(x)
  n <- length(x)
  d <- as.data.frame(helmert(n))
  d[[var]] <- rev(x)
  names(d) <- c(paste0(var, 1:(n-1)), var)
  d
}

#genres encoding
d <- encode_helmert(edx, "genres")

d <- d %>% mutate(genres_encoded = rowSums(d[,-797]))
```

```
#verifying that all items have unique encoding for genres column
length(unique(d$genres_encoded))

#merging
edx <- inner_join(edx,d[,c("genres","genres_encoded")], by= "genres")

#title encoding
d <- encode_helmert(edx, "title")

d <- d %>% mutate(title_encoded = rowSums(d[,-10676]))

#verifying that all items have unique encoding for title column
length(unique(d$title_encoded))

#merging
edx <- inner_join(edx,d[,c("title","title_encoded")], by= "title")
)
```

```
head(edx)
```

```
##    userId movieId rating timestamp                        title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                          genres release_year userId_stand movieId_stand
## 1:               Comedy|Romance         1992            0   0.001857766
## 2:         Action|Crime|Thriller         1995            0   0.002825032
## 3:  Action|Drama|Sci-Fi|Thriller         1995            0   0.004467850
## 4:       Action|Adventure|Sci-Fi         1994            0   0.004836332
## 5: Action|Adventure|Drama|Sci-Fi         1994            0   0.005035927
## 6:       Children|Comedy|Fantasy         1994            0   0.005435116
##    timestamp_stand genres_encoded_stand title_encoded_stand release_year_stand
## 1:       0.1117447          0.000000000        0.000000e+00          0.8279570
## 2:       0.1117413          0.001256281        9.367681e-05          0.8602151
## 3:       0.1117411          0.002512563        1.873536e-04          0.8602151
## 4:       0.1117410          0.003768844        2.810304e-04          0.8494624
## 5:       0.1117410          0.005025126        3.747073e-04          0.8494624
## 6:       0.1117434          0.006281407        4.683841e-04          0.8494624
##     u_effect  m_effect     ry_effect     gr_effect
## 1: -1.487535 0.5702365 -6.484236e-17  5.334526e-17
## 2: -1.487535 0.4057030  2.805265e-17 -2.855516e-16
## 3: -1.487535 0.1220525  2.805265e-17 -1.305178e-16
## 4: -1.487535 0.1971302  2.179846e-16  2.284709e-17
## 5: -1.487535 0.2103457  2.179846e-16 -2.241283e-16
## 6: -1.487535 0.9481829  2.179846e-16  5.484596e-17
```

Preprocessing: converting all the numeric variables to range between 0 and 1

```
#transforming the data for numeric predictors only
preProcess_range_model <- preProcess(edx[,-c("rating","genres","title")], method='range')

new_edx <- predict(preProcess_range_model, newdata = edx[,-c("rating","genres","title")])

#verifying that limits of potential predictors are between 0 and 1
apply(new_edx, 2, FUN=function(x){c('min'=min(x), 'max'=max(x))})

#exploring the new and old edx datasets
head(new_edx)
head(edx)

#ensuring that there is no alteration of unique items
length(unique(new_edx$userId))
length(unique(edx$userId))

#appending ratings and character variables
edx$userId_stand <- new_edx$userId
edx$movieId_stand <- new_edx$movieId
edx$timestamp_stand <- new_edx$timestamp
edx$genres_encoded_stand <- new_edx$genres_encoded
edx$title_encoded_stand <- new_edx$title_encoded
edx$release_year_stand <- new_edx$release_year

#removing new_edx and redundant columns in edx to preserve memory
rm(new_edx)
edx <- edx[,-c("genres_encoded", "title_encoded" )]
```

```
#exploring the amended edx data set
head(edx[,1:13])
```

```
##    userId movieId rating timestamp                      title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                          genres release_year userId_stand movieId_stand
## 1:               Comedy|Romance         1992            0   0.001857766
## 2:         Action|Crime|Thriller         1995            0   0.002825032
## 3:  Action|Drama|Sci-Fi|Thriller         1995            0   0.004467850
## 4:       Action|Adventure|Sci-Fi         1994            0   0.004836332
## 5: Action|Adventure|Drama|Sci-Fi         1994            0   0.005035927
## 6:       Children|Comedy|Fantasy         1994            0   0.005435116
##    timestamp_stand genres_encoded_stand title_encoded_stand release_year_stand
## 1:       0.1117447          0.000000000        0.000000e+00          0.8279570
## 2:       0.1117413          0.001256281        9.367681e-05          0.8602151
## 3:       0.1117411          0.002512563        1.873536e-04          0.8602151
## 4:       0.1117410          0.003768844        2.810304e-04          0.8494624
```

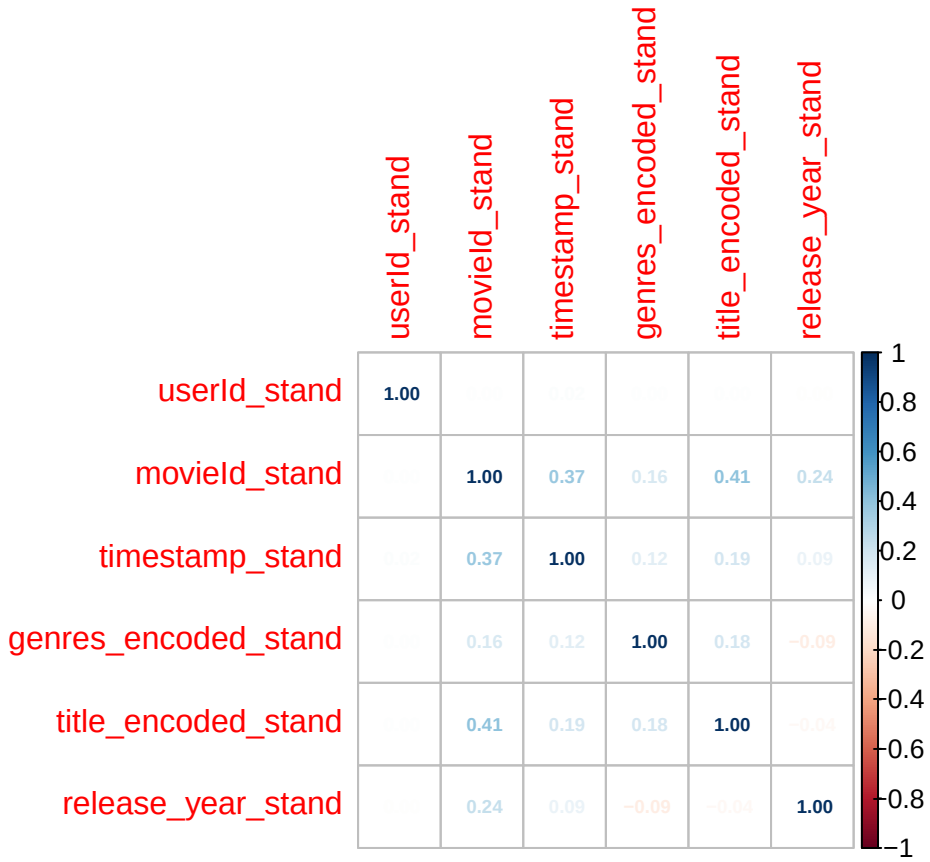
```
## 5:       0.1117410          0.005025126       3.747073e-04          0.8494624
## 6:       0.1117434          0.006281407       4.683841e-04          0.8494624
```

Visualising the improtance of variables

The following plot shows the correlation between potential predictors:

```
cor_edx <- edx[,8:13] %>% cor()
```

```
corrplot(cor_edx, method="number",type= "full",insig = "blank", number.cex = 0.6)
```



The above graphic seems to indicate that most predictors are independent. A positive cor relation between MovieId and title_encoded is noted. This is logical as both the movieId and title represent the particular movie. Therefore, one of these variables could be excluded from the model. The release year is also slightly correlated to the movieId, which is a logical association.

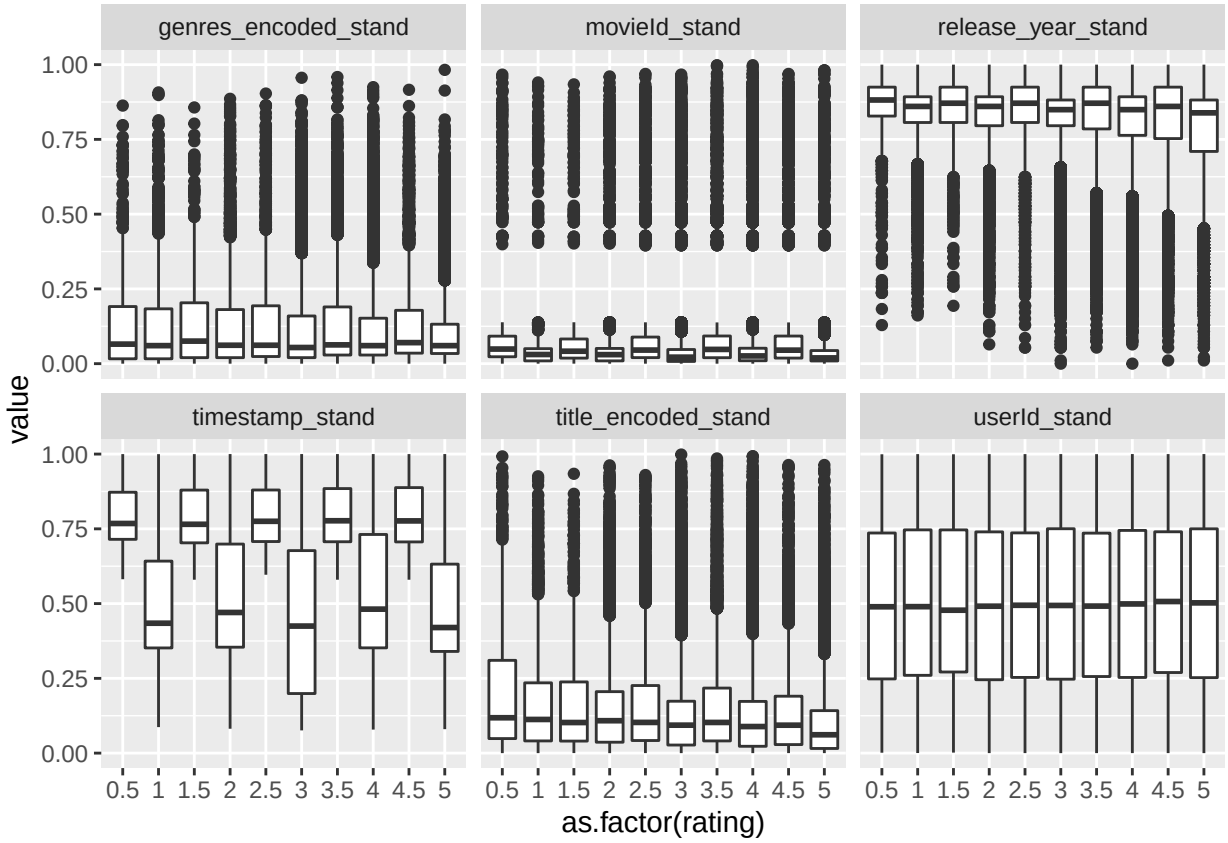
There is a slight positive correlation between movieId and timestamp, but there is no indication of causality.

- Variability:

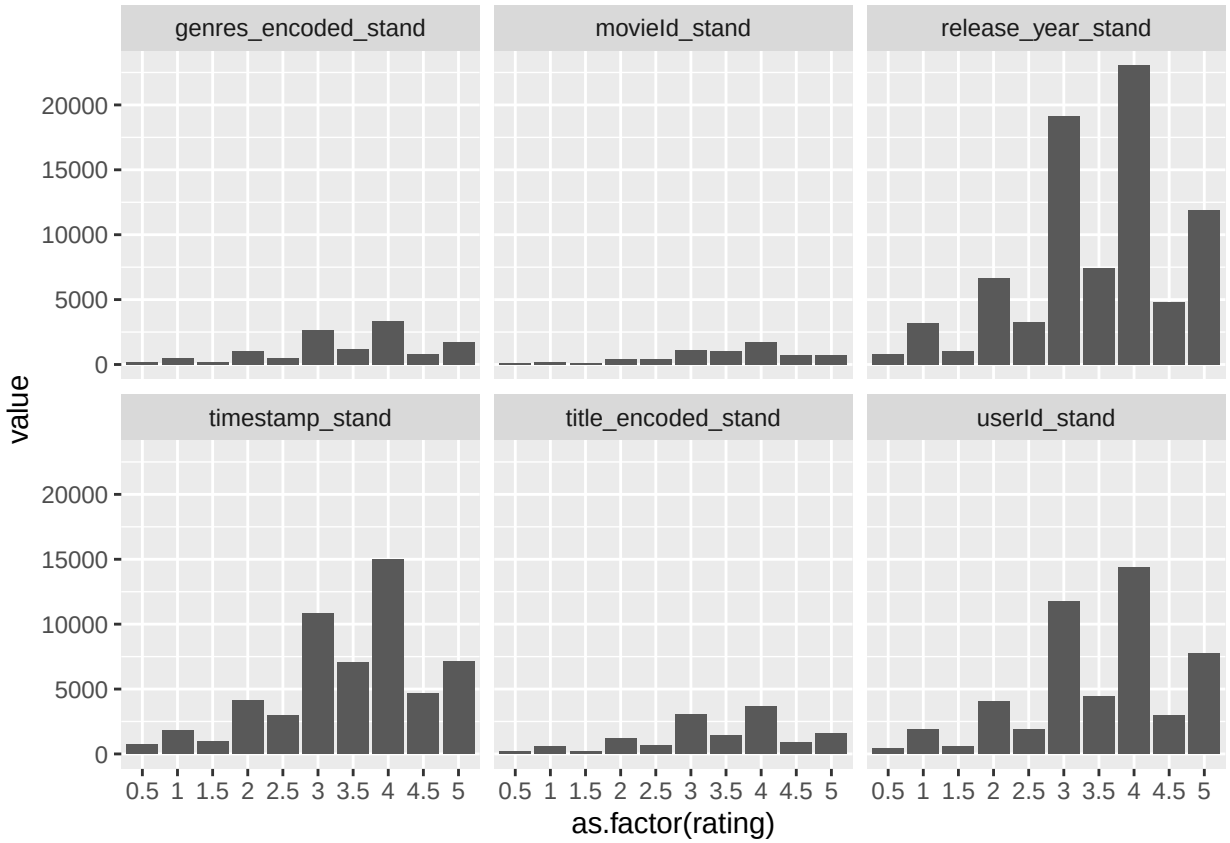
I intended to identify variability within the various predictors when grouped by rating category. A box plot comparison helps to identify changes in the mean between categories for each potential predictor, as an indicator that the predictor could have a significant effect in predicting the rating. A bar plot helps visualising the variability among categories for each predictor.

Due to large data size and to aid processing time, I have extracted random samples of the pre-processed data and plotted it using the rating field as a categorical variable.

```
sample_n(edx[,c(8:13,3)],100000) %>%
  pivot_longer( userId_stand:release_year_stand ) %>%
  group_by(as.factor(rating)) %>%
  filter(n()>=100) %>% ggplot() +
  geom_boxplot(aes(as.factor(rating),value)) +
  facet_wrap(vars(name))
```




```
sample_n(edx[,c(8:13,3)],100000) %>%
  pivot_longer( userId_stand:release_year_stand ) %>%
  group_by(as.factor(rating)) %>%
  filter(n()>=100) %>% ggplot() +
  geom_col(aes(as.factor(rating),value)) +
  facet_wrap(vars(name))
```



From the above charts, the following was noted:

- An unusual variation for ratings with half star was noted in relation to those with full stars (e.g. 0.5 vs 1 star).
- From the box plot it doesn't appear that significant variability in the means exist for the remaining predictors. However, looking at the distribution of the ratings in the bar plots: user effect, genre effect and title effect may exist.
- MovieId and title distributions are not the same. This might indicate a distortion in the data coming from the encoding pre-processing step. However, it might also indicate an effect of words within the titles that may influence users rating decisions.

Quantifying variance influence and ranking variables

To evaluate the importance of the variables I performed an analysis of the effect of each variable over the variation between average rating and rating grouped by each variable.

```
#user effect
length(unique(edx$userId))

user_effect <-edx %>% group_by(userId) %>%
  summarise( mean_uId = mean(rating),
             u_effect = mean(mu_hat - mean_uId ))

edx <- left_join(edx, user_effect[,c(1,3)])
```

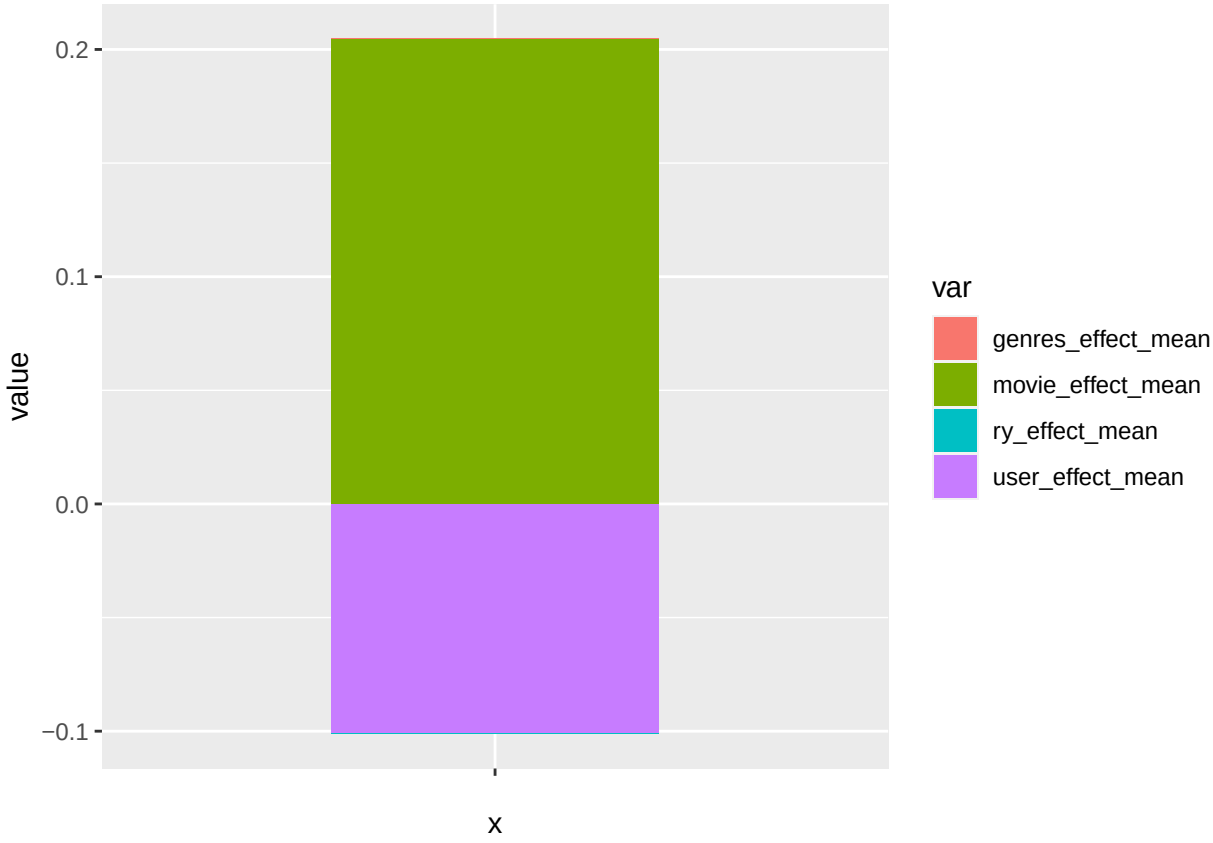
```
ranking_vars <- data.frame( var = "user_effect_mean" , value= mean(user_effect$u_effect))

rm(user_effect)

#movie effect (using movie ID  discarding title as its correlated)
length(unique(edx$movieId))

movie_effect <-edx %>% group_by(movieId) %>%
  summarize( mean_mId = mean(rating),
             m_effect = mean(mu_hat - mean_mId - u_effect))

edx <- left_join(edx, movie_effect[,c(1,3)])

ranking_vars <- bind_rows(ranking_vars,
                          data_frame(var = "movie_effect_mean" ,
                                     value= mean(movie_effect$m_effect)))

rm(movie_effect)

#released year effect
length(unique(edx$release_year))

ry_effect <-edx %>% group_by(release_year) %>%
  summarize( mean_ryId = mean(rating),
             ry_effect = mean(mu_hat- mean_ryId - u_effect  - m_effect ))

edx <- left_join(edx, ry_effect[,c(1,3)])

ranking_vars <- bind_rows(ranking_vars,
                          data_frame(var = "ry_effect_mean" ,
                                     value= mean(ry_effect$ry_effect)))

rm(ry_effect)

#genres year effect
length(unique(edx$genres))

genres_effect <-edx %>% group_by(genres) %>%
  summarize( mean_grId = mean(rating),
             gr_effect = mean(mu_hat- mean_grId - u_effect  - m_effect - ry_effect))

edx <- left_join(edx, genres_effect[,c(1,3)])

ranking_vars <- bind_rows(ranking_vars,
                          data_frame(var = "genres_effect_mean" ,
                                     value= mean(genres_effect$gr_effect)))

rm(genres_effect)

#timestamp effect not considered due to high carnality and no rational connection to ranking decision
length(unique(edx$timestamp))
```

- visualising the relevance of the predictors

```
ranking_vars
```

```
##                  var         value
## 1   user_effect_mean -1.011364e-01
## 2     ry_effect_mean -1.413301e-17
## 3 genres_effect_mean  1.044910e-17
## 4  movie_effect_mean  2.046178e-01
```

```
ranking_vars %>% ggplot(aes(x= "" , y=value, fill=var)) +  geom_bar (stat= "identity" , width=.5)
```



The above plot indicates that:

- Movie and user effects are the most significant predictors in the variation from the average rating.
- Released year and genres have a lower level of influence.
- Please note that timestamp field was not considered due to high cardinality and no rational connecting the rating decision to the time when the rating was placed.

Model fitting

Linear model

The above analysis indicates that the following predictors are relevant:

- userId
- movieId

Exploratory analysis above indicated that there were movies with low number of ratings , while other movies were rated only once. There were also very active users who rated several movies, while other less active rated only few movies.

This indicate a level of variability within these parameters that will cause a distortion in the estimated effects given that the mean estimated rated per movie and/or per user is not large enough to be reliable. Therefore,a regularisation technique is needed to fine tune the predictions, by penalising those predictions where the number of ratings is too low (and therefore the distortion between actual and predicted is larger) .

The linear model construct can be expressed as follows:

y_hat = mu + m_effect + u_effect + other

where:

- y_hat = predicted rating
- mu = non-conditional average rating across the full data set
- m_effect = the portion of the variation between the mu and the actual rating that can be explained by the movieId predictor
- u_effect = the portion of the variation between the mu and the actual rating that can be explained by the movieId predictor
- other = the portion of the variation between the mu and the actual rating where the cause is unidentified (i.e. noise)

The regularisation technique builds on refining the estimation of the root squared mean error as follows:

- Normal RMSE:

$\sum_{(y-mu-m_{(}effect_{)}-u_{(}effect_{)})^2}$

- With Regularisation

$\frac{1}{N}(\sum_{(y-mu-m_{(}effect_{)}-u_{(}effect_{)})^2} + \lambda(\sum_{b_m^2} + \sum_{b_u^2}))$

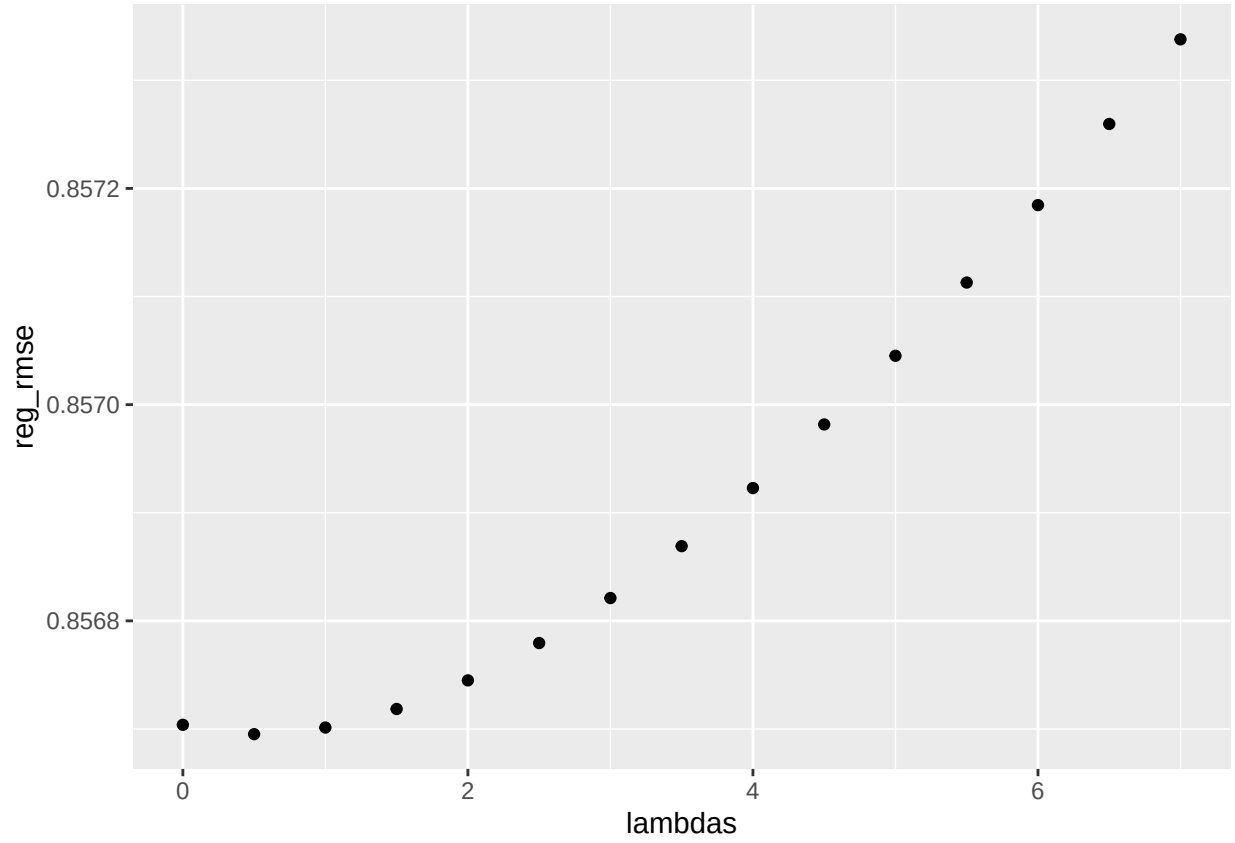
Where:

- N = number of actual ratings
- y = actual rating
- mu = non-conditional average rating across the full data set
- m_effect = the portion of the variation between the mu and the actual rating that can be explained by the movieId predictor
- u_effect = the portion of the variation between the mu and the actual rating that can be explained by the movieId predictor
- λ = optimised adjusting factor linked to the number of ratings for the specific movies/user
- b = the numbers of rating associated to a specific movie (m) or user (u)

```
lambdas <- seq(0, 7, 0.5)

reg_rmse <- sapply( lambdas, function(l){
  mu_hat <- mean(edx$rating)
  m <- edx %>%
    group_by(movieId) %>%
    summarize(reg_m_effect = sum(rating - mu_hat)/(n()+l))
  u <- edx %>%
    left_join(m, by="movieId") %>%
    group_by(userId) %>%
    summarize(reg_u_effect = sum(rating - reg_m_effect - mu_hat)/(n()+l))
  predicted_ratings <-
    edx %>%
    left_join(m, by = "movieId") %>%
    left_join(u, by = "userId") %>%
    mutate(mu = mean(rating), y_hat = mu + reg_m_effect + reg_u_effect) %>%
    .$y_hat
  return(RMSE(predicted_ratings, edx$rating))
})

lambda <- lambdas[which.min(reg_rmse)]
```

```
qplot(lambdas, reg_rmse)
```

optimising the regularisation parameter




```
lambda
```

```
## [1] 0.5
```

```
reg_rmse_val <- lapply( lambda, function(l){
  mu_hat <- mean(validation$rating)
  m <- validation %>%
    group_by(movieId) %>%
    summarize(reg_m_effect = sum(rating - mu_hat)/(n()+l))
  u <- validation %>%
    left_join(m, by="movieId") %>%
    group_by(userId) %>%
    summarize(reg_u_effect = sum(rating - reg_m_effect - mu_hat)/(n()+l))
  predicted_ratings <-
    validation %>%
    left_join(m, by = "movieId") %>%
    left_join(u, by = "userId") %>%
    mutate(mu = mean(rating), y_hat = mu + reg_m_effect + reg_u_effect) %>%
    .$y_hat
  return(RMSE(predicted_ratings, validation$rating))
})

reglm_naive <- 1 - ((as.numeric(reg_rmse_val)*(1-acc_guessing))/guess_rmse)
```

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularised_Lineal_Model",
                                     RMSE = as.numeric(reg_rmse_val), approx_accuracy= reglm_naive ))
```

predicting with regularised lineal model

Evaluating other possible models

I performed a quick evaluation of other models using a random sample of 1000 records to determine the following:

- Feasibility of use given run time
- Potential RMSE results (based on the single sample, this estimating could have been improved using a montecarlo simulation if the model would have been considered appropriate)

Preparing the validation dataset

Before evaluating models with the caret package, i prepared the validation data set as follows:

```
#preparing the validation set: convert relevant numeric variables to range between 0 and 1

preProcess_range_model_val <- preProcess(validation[,-c("rating","genres","title")], method='range')
new_val <- predict(preProcess_range_model_val, newdata = validation[,-c("rating","genres","title")])

#verifying that limits of potential predictors are between 0 and 1
apply(new_val, 2, FUN=function(x){c('min'=min(x), 'max'=max(x))})

#appending standarised fields
validation$userId_stand <- new_val$userId
validation$movieId_stand <- new_val$movieId

#exploding amended validation
head(validation)

rm(new_val)
```

Considering the potential models

I selected the following three (3) models that can be used for regression:

- Bayesian Regularized Neural Networks (brnn)
- Classification and Regression Trees (cart)
- Bayesian Ridge Regression [Model Averaged] (brr)

I used the following code to estimate the run-time based on a sample of 1000 records:

```r
# setting the control parameters
fitControl <- trainControl(
  method = 'cv',
  number = 2,
  savePredictions = 'final',
    )


# measuring run time for various models


#Bayesian Regularized Neural Networks (method = 'brnn')

brnn_grid <-  expand.grid(neurons = c(2, 3, 4))

set.seed(100)

startTime <- Sys.time()
model_brnn = train(rating ~ userId_stand + movieId_stand , data=sample_n(edx,1000), method='brnn', metri
                   tuneGrid = brnn_grid, trControl = fitControl)
endTime <- Sys.time()

print(endTime - startTime)

startTime <- Sys.time()
predict(model_brnn, sample_n(validation,1000))
endTime <- Sys.time()
print(endTime - startTime)


#CART (method = 'rpart2')

cart_grid <-  expand.grid(maxdepth = c(2, 3, 4, 5))

set.seed(100)

startTime <- Sys.time()
model_cart = train(rating ~ userId_stand + movieId_stand , data=sample_n(edx,1000), method='rpart2',
                   metric='RMSE',
                   tuneGrid = cart_grid,
                   trControl = fitControl)
endTime <- Sys.time()

print(endTime - startTime)

startTime <- Sys.time()
predict(model_cart, sample_n(validation,1000))
endTime <- Sys.time()
print(endTime - startTime)


# Bayesian Ridge Regression (Model Averaged)   method = 'blassoAveraged'
```

```
set.seed(100)

startTime <- Sys.time()
model_brr = train(rating ~ userId_stand + movieId_stand , data=sample_n(edx,1000),
                  method='blassoAveraged', metric='RMSE',
                  trControl = fitControl)
endTime <- Sys.time()

print(endTime - startTime)

startTime <- Sys.time()
predict(model_brr, sample_n(validation,1000))
endTime <- Sys.time()
print(endTime - startTime)

#comparing sample models

sample_models_compare <- resamples(list(brnn = model_brnn,
                                        cart = model_cart,
                                        brr = model_brr))
```

The estimated run-time in the full population was considered significant.

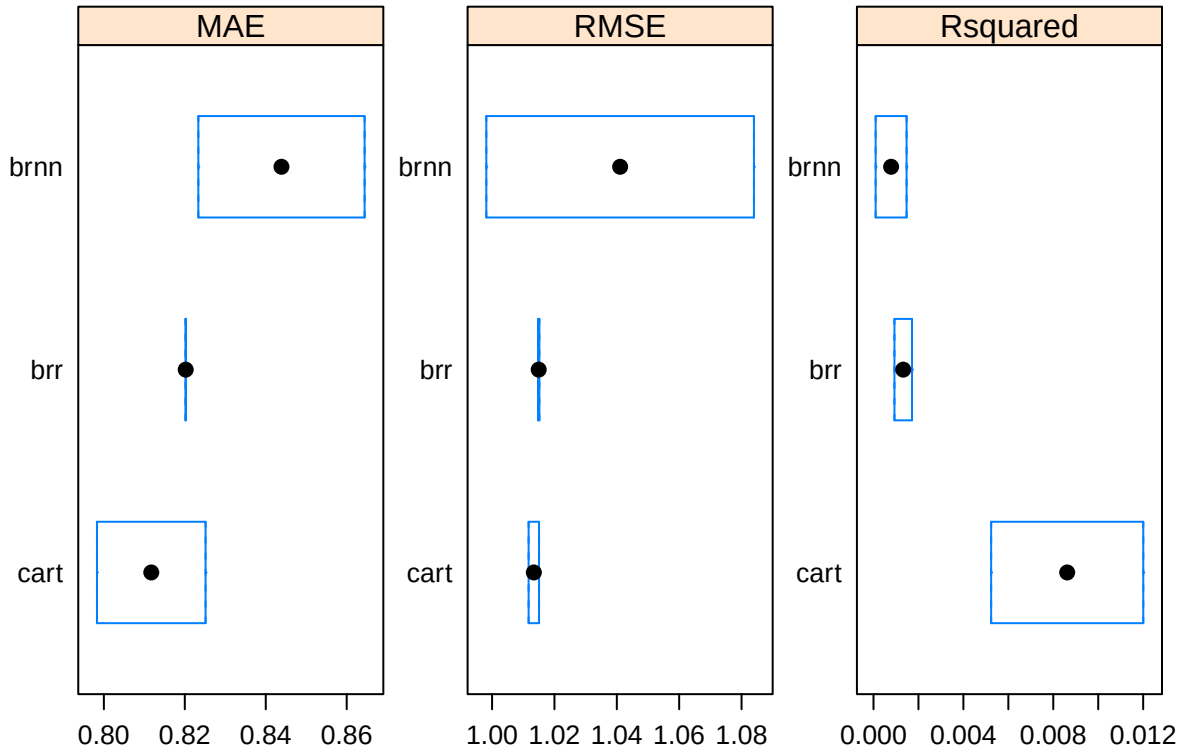
The following code shows the results of comparing the tested models

```
# Summary of the sample models performances
summary(sample_models_compare)
```

```
##
## Call:
## summary.resamples(object = sample_models_compare)
##
## Models: brnn, cart, brr
## Number of resamples: 2
##
## MAE
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## brnn 0.8233437 0.8336138 0.8438838 0.8438838 0.8541538 0.8644238    0
## cart 0.7983043 0.8050116 0.8117189 0.8117189 0.8184262 0.8251335    0
## brr  0.8201530 0.8201880 0.8202230 0.8202230 0.8202581 0.8202931    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## brnn 0.9981535 1.019609 1.041064 1.041064 1.062519 1.083974    0
## cart 1.0117172 1.012553 1.013389 1.013389 1.014225 1.015061    0
## brr  1.0147316 1.014838 1.014944 1.014944 1.015050 1.015157    0
##
## Rsquared
##              Min.      1st Qu.       Median         Mean     3rd Qu.
## brnn 9.615633e-05 0.0004401738 0.0007841913 0.0007841913 0.001128209
## cart 5.234525e-03 0.0069269131 0.0086193013 0.0086193013 0.010311689
## brr  9.319251e-04 0.0011274869 0.0013230487 0.0013230487 0.001518610
##             Max. NA's
```

```
## brnn 0.001472226    0
## cart 0.012004078    0
## brr  0.001714172    0
```

```
# Using box plots to compare models
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(sample_models_compare, scales=scales)
```



Given the significant run-time needed to apply the above models and the indication that there may not be an improvement in the RMSE in comparison to the Regularised Lineal Model, I decided not to implement these alternative models.

Reference

The following material was used a reference:

- EDX course material
- Caret Package – A Practical Guide to Machine Learning in R
- Helmert Encoding: details on Helmert encoding can be found in this StackOverlow answer by user StatsStudent and the comments by user whuber.