Movie recommendation results

Loading packages and data

```r
# Load required packages
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.1       v purrr   0.3.2
## v tibble  2.1.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
## Warning: package 'tibble' was built under R version 3.5.3
```

```
## Warning: package 'tidyr' was built under R version 3.5.3
```

```
## Warning: package 'readr' was built under R version 3.5.3
```

```
## Warning: package 'purrr' was built under R version 3.5.3
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

```
## Warning: package 'stringr' was built under R version 3.5.3
```

```
## Warning: package 'forcats' was built under R version 3.5.3
```

```
## -- Conflicts ------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```r
# Load locally stored data
edx_original <- readRDS("edx.RData")
validation_original <- readRDS("validation.RData")
```

Preparing data

```
# Separating user characteristics, ratings and movie features
user_data <- edx_original %>% select(userId) %>%
                              mutate(userId = as.factor(userId)) %>% unique()

rating_data <- edx_original %>% select(userId, movieId, rating, timestamp) %>%
                                mutate(userId = as.factor(userId),
                                       movieId = as.factor(movieId),
                                       date = as.POSIXct(timestamp, origin = "1970-01-01")) %>% # Turn
                                mutate(rating_year = year(date)) %>%
                                select(-timestamp, -date)

# MOVIE DATA
movie_data <- edx_original %>% select(movieId, title, genres) %>%
                               unique() %>%
                               mutate(movieId = as.factor(movieId),
                                      movie_year = as.factor(str_sub(title, -5, -2)),
                                      title = str_sub(title, 0, -8))

# Let's determine which genres we have
unique_genres <- unique(str_extract(movie_data$genres, "[^\\|]+"))
# Now lets add them as columns
movie_data[, unique_genres] <- NA
# For each of the ratings we detect present genres
for(g in 1:length(unique_genres)) {
  #print(paste("Detecting: ", unique_genres[g]))
  movie_data[, unique_genres[g]] <- str_detect(movie_data$genres, unique_genres[g])
}
# Drop old columns
movie_data <- movie_data %>% select(-genres)

# Let's explore our new data sets
head(user_data)
```

```
##     userId
## 1        1
## 20       2
## 37       3
## 68       4
## 103      5
## 177      6
```

```
head(rating_data)
```

```
##   userId movieId rating rating_year
## 1      1     122      5        1996
## 2      1     185      5        1996
## 3      1     292      5        1996
## 4      1     316      5        1996
## 5      1     329      5        1996
## 6      1     355      5        1996
```

```r
head(movie_data)
```

```
##   movieId                  title movie_year Comedy Action Children
## 1     122              Boomerang       1992   TRUE  FALSE    FALSE
## 2     185               Net, The       1995  FALSE   TRUE    FALSE
## 3     292               Outbreak       1995  FALSE   TRUE    FALSE
## 4     316               Stargate       1994  FALSE   TRUE    FALSE
## 5     329 Star Trek: Generations       1994  FALSE   TRUE    FALSE
## 6     355       Flintstones, The       1994   TRUE  FALSE     TRUE
##   Adventure Animation Drama Crime Sci-Fi Horror Thriller Film-Noir Mystery
## 1     FALSE     FALSE FALSE FALSE  FALSE  FALSE    FALSE     FALSE   FALSE
## 2     FALSE     FALSE FALSE  TRUE  FALSE  FALSE     TRUE     FALSE   FALSE
## 3     FALSE     FALSE  TRUE FALSE   TRUE  FALSE     TRUE     FALSE   FALSE
## 4      TRUE     FALSE FALSE FALSE   TRUE  FALSE    FALSE     FALSE   FALSE
## 5      TRUE     FALSE  TRUE FALSE   TRUE  FALSE    FALSE     FALSE   FALSE
## 6     FALSE     FALSE FALSE FALSE  FALSE  FALSE    FALSE     FALSE   FALSE
##   Western Documentary Romance Fantasy Musical   War  IMAX
## 1   FALSE       FALSE    TRUE   FALSE   FALSE FALSE FALSE
## 2   FALSE       FALSE   FALSE   FALSE   FALSE FALSE FALSE
## 3   FALSE       FALSE   FALSE   FALSE   FALSE FALSE FALSE
## 4   FALSE       FALSE   FALSE   FALSE   FALSE FALSE FALSE
## 5   FALSE       FALSE   FALSE   FALSE   FALSE FALSE FALSE
## 6   FALSE       FALSE   FALSE    TRUE   FALSE FALSE FALSE
##   (no genres listed)
## 1              FALSE
## 2              FALSE
## 3              FALSE
## 4              FALSE
## 5              FALSE
## 6              FALSE
```

User characteristics

```r
## Ratings statistics
user_stats <- rating_data %>% group_by(userId) %>%
                                summarise(user_ratings = n(),
                                          user_median = median(rating),
                                          user_mean = mean(rating),
                                          user_sd = sd(rating)) %>%
                                select(userId, user_median, user_mean, user_sd) %>%
                                unique()

user_data <- user_data %>% inner_join(user_stats, by = "userId")


## TO DO: ADD GENRE AVERAGES
```

Movie characteristics

```
# Movie rating statistics
movie_stats <- rating_data %>% group_by(movieId) %>%
                               summarise(movie_ratings = n(),
                                         movie_median = median(rating),
                                         movie_mean = mean(rating),
                                         movie_sd = sd(rating)) %>%
                               select(movieId, movie_median, movie_mean, movie_sd) %>%
                               unique()

movie_data <- movie_data %>% inner_join(movie_stats, by = "movieId")
```

Rating characteristics

```
## Rating characteristics
rating_mean <- mean(rating_data$rating)

rating_data <- rating_data %>% mutate(rating_mean_diff = rating - rating_mean) %>%
                               inner_join(movie_data[, c("movieId", "movie_mean")], by = "movieId") %>%
                               mutate(movie_mean_diff = rating - movie_mean) %>%
                               inner_join(user_data[, c("userId", "user_mean")], by = "userId") %>%
                               mutate(user_mean_diff = rating - user_mean)
```

Model training preparation

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
# RSME loss function function to evaluate models
RMSE <- function(true_ratings, predicted_ratings){
     sqrt(mean((true_ratings - predicted_ratings)^2))
}

# Create train and test sets
```

```r
test_index <- createDataPartition(y = rating_data$rating, times = 1,
                                  p = 0.2, list = FALSE)
train_set <- rating_data[-test_index,]
test_set <- rating_data[test_index,]

# To make sure movies and users are in both train and test sets
test_set <- test_set %>%
     semi_join(train_set, by = "movieId") %>%
     semi_join(train_set, by = "userId")
```

Manual model building

```r
### Building the Recommendation System

# Naive model using average rating
rating_mean <- mean(train_set$rating)
rating_mean
```

```
## [1] 3.51242
```

```r
naive_rmse <- RMSE(test_set$rating, rating_mean)

rmse_results <- data_frame(method = "Average rating", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```r
# Model using fixed number
fixed_number <- rep(2.5, nrow(test_set))

fixed_rmse <- RMSE(test_set$rating, fixed_number)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "Fixed number 2.5",
                                     RMSE = fixed_rmse ))

rmse_results
```

```
## # A tibble: 2 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Average rating    1.06
## 2 Fixed number 2.5  1.47
```

Derived models

```
# Derived predictions

predictions <- test_set %>% group_by(movieId) %>%
               # Rating effect (effect of rating scale used; looking at differences from rating mean)
               summarise(rating_mean_diff_avg = rating_mean + mean(rating_mean_diff),
               # Movie effect (effect of general movie popularity; looking at differences between user
                         movie_mean_diff_avg = rating_mean + mean(movie_mean_diff),
               # User effect (effect of user rating behaviour; looking at differeces between rating and
                         user_mean_diff_avg = rating_mean + mean(user_mean_diff),
               # User-movie effect (effect of user rating behaviour; looking at differeces between user
                         user_movie_mean_diff_avg = mean(movie_mean) + mean(user_mean_diff))

# Calculate RMSEs

rating_effect_rmse <- RMSE(test_set$rating,
                           predictions$rating_mean_diff_avg)
```

```
## Warning in true_ratings - predicted_ratings: longer object length is not a
## multiple of shorter object length
```

```
movie_effect_rmse <- RMSE(test_set$rating,
                          predictions$movie_mean_diff_avg)
```

```
## Warning in true_ratings - predicted_ratings: longer object length is not a
## multiple of shorter object length
```

```
user_effect_rmse <- RMSE(test_set$rating,
                         predictions$user_mean_diff_avg)
```

```
## Warning in true_ratings - predicted_ratings: longer object length is not a
## multiple of shorter object length
```

```
user_movie_effect_rmse <- RMSE(test_set$rating,
                               predictions$user_movie_mean_diff_avg)
```

```
## Warning in true_ratings - predicted_ratings: longer object length is not a
## multiple of shorter object length
```

```
# Print RMSEs

rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "Rating effect",
                                     RMSE = rating_effect_rmse))
rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "Movie effect",
                                     RMSE = movie_effect_rmse))
rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "User effect",
                                     RMSE = user_effect_rmse))
rmse_results <- bind_rows(rmse_results,
```

```r
                          data_frame(method = "User-movie effect",
                                     RMSE = user_movie_effect_rmse))
rmse_results
```

```
## # A tibble: 6 x 2
##   method              RMSE
##   <chr>              <dbl>
## 1 Average rating      1.06
## 2 Fixed number 2.5    1.47
## 3 Rating effect       1.27
## 4 Movie effect        1.11
## 5 User effect         1.21
## 6 User-movie effect   1.59
```

Merge all data for calculating models

```r
# Let's merged derived data
train_set_plus <- train_set %>% select(-user_mean) %>%
                                left_join(user_data, by = "userId") %>%
                                select(-movie_mean) %>%
                                left_join(movie_data, by = "movieId")

test_set_plus <- test_set %>% select(-user_mean) %>%
                                left_join(user_data, by = "userId") %>%
                                select(-movie_mean) %>%
                                left_join(movie_data, by = "movieId")
```

Train calculated models

```r
# Let's prepare parallel processing
library(doParallel)
```

```
## Warning: package 'doParallel' was built under R version 3.5.3

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loading required package: iterators

## Warning: package 'iterators' was built under R version 3.5.1
```

```
## Loading required package: parallel
```

```r
cl <- makePSOCKcluster(5)
registerDoParallel(cl)

## Model training in parallel

tc <- trainControl(number = 3)
training_data <- train_set_plus[1:100000,]

system.time({
  model_1 <- train(rating ~ movie_median,
                   data = training_data, method = "lm",
                   na.action = na.omit, metric = "RMSE",
                   trainControl = tc)
})
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trainControl' will be disregarded

##    user  system elapsed
##    1.53    0.05    9.18
```

```r
model_1$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat, trainControl = ..1)
##
## Coefficients:
##  (Intercept)  movie_median
##       0.6499        0.8104
```

```r
system.time({
  model_2 <- train(rating ~ movie_median + movie_mean,
                   data = training_data, method = "lm",
                   na.action = na.omit, metric = "RMSE",
                   trainControl = tc)
})
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trainControl' will be disregarded

##    user  system elapsed
##    1.56    0.11    3.20
```

```r
model_2$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat, trainControl = ..1)
```

```
##
## Coefficients:
##  (Intercept)  movie_median    movie_mean
##     -0.01194      -0.01049       1.02753
```

```r
system.time({
  model_3 <- train(rating ~ movie_median + movie_mean + movie_sd,
                   data = training_data, method = "lm",
                   na.action = na.omit, metric = "RMSE",
                   trainControl = tc)
})
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trainControl' will be disregarded

##    user  system elapsed
##    1.60    0.09    2.93
```

```r
model_3$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat, trainControl = ..1)
##
## Coefficients:
##  (Intercept)  movie_median    movie_mean      movie_sd
##     -0.09988      -0.01362       1.03891       0.06294
```

```r
## Does it matter what the movies are? Maybe focus on the ratings

system.time({
  model_4 <- train(rating ~ user_median,
                   data = training_data, method = "lm",
                   na.action = na.omit, metric = "RMSE",
                   trainControl = tc)
})
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trainControl' will be disregarded

##    user  system elapsed
##    1.55    0.08    2.87
```

```r
model_4$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat, trainControl = ..1)
##
## Coefficients:
## (Intercept)  user_median
##      1.0848       0.6759
```

```r
system.time({
  model_5 <- train(rating ~ user_median + user_mean,
                   data = training_data, method = "lm",
                   na.action = na.omit, metric = "RMSE",
                   trainControl = tc)
})
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trainControl' will be disregarded

##    user  system elapsed
##    1.56    0.11    2.70
```

```r
model_5$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat, trainControl = ..1)
##
## Coefficients:
## (Intercept)  user_median    user_mean
##   0.0007519   -0.0054273    1.0053696
```

```r
system.time({
  model_6 <- train(rating ~ user_median + user_mean + user_sd,
                   data = training_data, method = "lm",
                   na.action = na.omit, metric = "RMSE",
                   trainControl = tc)
})
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trainControl' will be disregarded

##    user  system elapsed
##    1.50    0.14    2.83
```

```r
model_6$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat, trainControl = ..1)
##
## Coefficients:
## (Intercept)  user_median    user_mean      user_sd
##    0.021604    -0.002668     1.000265    -0.013519
```

```r
# User and movie effects combined

system.time({
  model_7 <- train(rating ~ movie_median + user_median,
```

```r
                   data = training_data, method = "lm",
                   na.action = na.omit, metric = "RMSE",
                   trainControl = tc)
})
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trainControl' will be disregarded

##    user  system elapsed
##    1.58    0.05    2.77
```

```r
model_7$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat, trainControl = ..1)
##
## Coefficients:
##  (Intercept)  movie_median   user_median
##      -1.2895        0.7458        0.5889
```

```r
system.time({
  model_8 <- train(rating ~ movie_median + movie_mean + user_median + user_mean,
                   data = training_data, method = "lm",
                   na.action = na.omit, metric = "RMSE",
                   trainControl = tc)
})
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trainControl' will be disregarded

##    user  system elapsed
##    1.68    0.08    3.33
```

```r
model_8$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat, trainControl = ..1)
##
## Coefficients:
##  (Intercept)  movie_median    movie_mean   user_median     user_mean
##   -2.6614750     0.0159165     0.8877218     0.0001111     0.8514821
```

```r
system.time({
  model_9 <- train(rating ~ movie_median + movie_mean + movie_sd + user_median + user_mean + user_sd,
                   data = training_data, method = "lm",
                   na.action = na.omit, metric = "RMSE",
                   trainControl = tc)
})
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trainControl' will be disregarded
```

```
##    user  system elapsed
##    1.76    0.04    3.42
```

```
model_9$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat, trainControl = ..1)
##
## Coefficients:
##  (Intercept)  movie_median    movie_mean      movie_sd   user_median
##    -2.665190      0.014477      0.892506      0.025176      0.004373
##    user_mean       user_sd
##     0.843633     -0.020355
```

```
# Basic movie_mean and user_mean seem most influential
final_model_data <- train_set_plus %>% select(movieId, rating, movie_mean, user_mean) %>% unique()
system.time({
  model_10 <- train(rating ~ movie_mean + user_mean,
                    data = final_model_data, method = "lm",
                    na.action = na.omit, metric = "RMSE",
                    trainControl = tc)
})
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trainControl' will be disregarded
```

```
##    user  system elapsed
##   48.75    5.78  170.15
```

```
model_10$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat, trainControl = ..1)
##
## Coefficients:
## (Intercept)   movie_mean    user_mean
##     -2.5280       0.8791       0.8390
```

```
## Stopping clusters
stopCluster(cl)
```

Prediction and results

```r
# Predict on test_set
model_1_predictions <- predict(model_1, newdata = test_set_plus)
model_2_predictions <- predict(model_2, newdata = test_set_plus)
model_3_predictions <- predict(model_3, newdata = test_set_plus)
model_4_predictions <- predict(model_4, newdata = test_set_plus)
model_5_predictions <- predict(model_5, newdata = test_set_plus)
model_6_predictions <- predict(model_6, newdata = test_set_plus)
model_7_predictions <- predict(model_7, newdata = test_set_plus)
model_8_predictions <- predict(model_8, newdata = test_set_plus)
model_9_predictions <- predict(model_9, newdata = test_set_plus)
model_10_predictions <- predict(model_10, newdata = test_set_plus)

# Print results on test set

print(paste("Model 1 RMSE in test set: ", RMSE(test_set_plus$rating, model_1_predictions)))
```

```
## [1] "Model 1 RMSE in test set:  0.961856858351706"
```

```r
print(paste("Model 2 RMSE in test set: ", RMSE(test_set_plus$rating, model_2_predictions)))
```

```
## [1] "Model 2 RMSE in test set:  0.943715761622832"
```

```r
print(paste("Model 3 RMSE in test set: ", RMSE(test_set_plus$rating, model_3_predictions)))
```

```
## [1] "Model 3 RMSE in test set:  0.943720720931866"
```

```r
print(paste("Model 4 RMSE in test set: ", RMSE(test_set_plus$rating, model_4_predictions)))
```

```
## [1] "Model 4 RMSE in test set:  0.992816763316907"
```

```r
print(paste("Model 5 RMSE in test set: ", RMSE(test_set_plus$rating, model_5_predictions)))
```

```
## [1] "Model 5 RMSE in test set:  0.970130044036678"
```

```r
print(paste("Model 6 RMSE in test set: ", RMSE(test_set_plus$rating, model_6_predictions)))
```

```
## [1] "Model 6 RMSE in test set:  0.97013451409216"
```

```r
print(paste("Model 7 RMSE in test set: ", RMSE(test_set_plus$rating, model_7_predictions)))
```

```
## [1] "Model 7 RMSE in test set:  0.9072658857676"
```

```r
print(paste("Model 8 RMSE in test set: ", RMSE(test_set_plus$rating, model_8_predictions)))
```

```
## [1] "Model 8 RMSE in test set:  0.8725047933572"
```

```r
print(paste("Model 9 RMSE in test set: ", RMSE(test_set_plus$rating, model_9_predictions)))
```

```
## [1] "Model 9 RMSE in test set:  0.872542123139385"
```

```r
print(paste("Model 10 RMSE in test set: ", RMSE(test_set_plus$rating, model_10_predictions)))
```

```
## [1] "Model 10 RMSE in test set:  0.8724270826599"
```

```r
# Verify results on validation set
```

```r
# Shall we explore auto ML?

library(h2o)
```

```
## Warning: package 'h2o' was built under R version 3.5.3
```

```
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit http://docs.h2o.ai
##
## ----------------------------------------------------------------------
```

```
##
## Attaching package: 'h2o'
```

```
## The following objects are masked from 'package:lubridate':
##
##     day, hour, month, week, year
```

```
## The following objects are masked from 'package:stats':
##
##     cor, sd, var
```

```
## The following objects are masked from 'package:base':
##
##     %*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc
```

```r
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         3 hours 25 minutes
##     H2O cluster timezone:       Europe/Berlin
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.22.1.1
##     H2O cluster version age:    4 months and 22 days !!!
##     H2O cluster name:           H2O_started_from_R_Codrin_wgs510
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   13.18 GB
##     H2O cluster total cores:    12
##     H2O cluster allowed cores:  12
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Algos, AutoML, Core V3, Core V4
##     R Version:                  R version 3.5.0 (2018-04-23)
```

```
## Warning in h2o.clusterInfo():
## Your H2O cluster version is too old (4 months and 22 days)!
## Please download and install the latest version from http://h2o.ai/download/
```

```r
# Import a sample binary outcome train/test set into H2O
train <- as.h2o(train_set_plus)
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```r
test <- as.h2o(test_set_plus)
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```r
# Identify predictors and response
y <- "rating"
x <- setdiff(names(train_set_plus), y)

# Run AutoML for 10 base models (limited to 1 hour max runtime by default)
aml <- h2o.automl(x = x, y = y,
                  training_frame = train,
                  max_models = 10,
                  seed = 1)
```

```
##
  |
  |                                                                 |   0%
  |
  |=                                                                |   1%
  |
  |=                                                                |   2%
  |
  |==                                                               |   2%
  |
  |==                                                               |   3%
  |
  |==                                                               |   4%
  |
  |===                                                              |   4%
  |
  |===                                                              |   5%
  |
  |====                                                             |   6%
  |
  |====                                                             |   7%
  |
  |=====                                                            |   7%
  |
  |=====                                                            |   8%
  |
  |======                                                           |   9%
  |
  |======                                                           |  10%
  |
  |=======                                                          |  10%
  |
  |=======                                                          |  11%
  |
  |========                                                         |  12%
  |
  |========                                                         |  13%
  |
  |=========                                                        |  13%
  |
  |=========                                                        |  14%
  |
  |==========                                                       |  15%
  |
  |==========                                                       |  16%
  |
  |===========                                                      |  16%
  |
  |===========                                                      |  17%
  |
  |===========                                                      |  18%
  |
  |============                                                     |  18%
  |
```

```
  |============                                                      |  19%
  |
  |=============                                                     |  20%
  |
  |==========================================================        |  88%
  |
  |==============================================================    |  94%
  |
  |==================================================================| 100%
```

```r
# View the AutoML Leaderboard
lb <- aml@leaderboard
print(lb, n = nrow(lb))
```

```
##                                              model_id
## 1                        DRF_1_AutoML_20190520_180116
## 2                        XRT_1_AutoML_20190520_180116
## 3    StackedEnsemble_AllModels_AutoML_20190520_180116
## 4 StackedEnsemble_BestOfFamily_AutoML_20190520_180116
## 5           GLM_grid_1_AutoML_20190520_180116_model_1
##   mean_residual_deviance         rmse          mse          mae
## 1           1.133491e-07 0.0003366736 1.133491e-07 8.194040e-06
## 2           5.458101e-07 0.0007387896 5.458101e-07 1.646689e-04
## 3           5.483918e-06 0.0023417767 5.483918e-06 1.858332e-03
## 4           5.483918e-06 0.0023417767 5.483918e-06 1.858332e-03
## 5           2.873683e-04 0.0169519415 2.873683e-04 1.359361e-02
##          rmsle
## 1 0.0001312058
## 2 0.0001985072
## 3 0.0007049445
## 4 0.0007049445
## 5 0.0044671316
##
## [5 rows x 6 columns]
```