MovieLens Report

Aditya Chate

05/01/2021

Overview:

The objective of the project comprised of creating a movie rating prediction system with the provided dataset. The "edx" and "validation" datasets were created with code provided in the project introduction on the course website. The "edx" dataset was used for data-exploration, analysis, and machine learning algorithm training purposes. With due consideration for computing capacity and computing time, a random sampling approach was employed. The predictions generated were tested against the "validation" set to obtain an **RMSE** (root mean square error) of **0.69**.

The following linear model was fit to the data:

$$y = mu + b\_i + b\_u + b\_y + b\_g + epsilon$$

where,
y = **predicted rating,**
mu = **average rating,**
b_i = **movie bias,**
b_u = **user bias,**
b_y = **movie release year bias,**
b_g = **genre bias,**
$epsilon$ = **independent errors from sampling from the same dataset.**

The above variables required for the linear model represent sources of variation in rating prediction, and hence are considered as "biases". They were either directly obtained from the data on variables present in the dataset, or were derived by wrangling and processing the available data. A regularization parameter, *lambda* = 3.25, was obtained using a representative sample of 10000 observations. This was used to regularize the predictions generated by an averaging approach. Readings for the "biases" and the predictions obtained by applying them one after the other, their regularized equivalents, and the average ratings for the movies were used together as a set of 17 predictors to train the machine learning models. Once again, with due consideration for computing time and computing capacity, six different sets of 1000 samples each were used. Each set was trained on nine different individual machine learning models, and one ensemble model. Rating prediction results of each of those were checked against actual ratings for the same movies in the validation dataset. The RMSEs showed a **standard deviation** of **0.022**, and all results (individual RMSEs) were within two standard deviations of the overall mean RMSE, thus indicating good reproducibility of results. The **minimum RMSE** obtained among all runs of the machine learning models was **0.65**, and the mean of all RMSEs, yielded a **final RMSE** of **0.69**.

Methodology and Analysis:

Definition of the RMSE(root mean square error) function:

The RMSE(root mean square error) function calculates the difference between the predicted value and the true (observed) value of a variable in a dataset which, in this case, is the rating of a movie by a specific user. The function involves taking a square of the difference to obtain a positive number and then taking the square root, thus giving the absolute value of the difference. The average of these values is taken across the dataset to give the RMSE of the dataset. If the difference between the predicted values and the actual observed values is minimal, then the RMSE function will output a minimum value, thus indicating the accuracy of the predictions. An RMSE of less than 0.86490 was expected for full points in the final result section of the project.

```
RMSE <- function(true_ratings, predicted_ratings)
{
sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Dataset Structures:

The "edx" and the "validation" datasets are both created from the MovieLens dataset using code provided on the course website. These are the datasets to be used for analysis and algorithm training, and for validating the generated predictions respectively. Examining the structure of the given datasets reveals each to be comprising of observations on six variables, namely, *userId*, *movieId*, *rating*, *timestamp*, *title*, and *genres*.

Structure of edx dataset for analysis and training of algorithms to generate predictions:

```
## Classes 'data.table' and 'data.frame':   6 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1
##  $ movieId  : num  122 185 292 316 329 355
##  $ rating   : num  5 5 5 5 5 5
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|A
##  - attr(*, ".internal.selfref")=<externalptr>
```

Structure of validation dataset for final RMSE evaluation:

```
## Classes 'data.table' and 'data.frame':   6 obs. of  6 variables:
##  $ userId   : int  1 1 1 2 2 2
##  $ movieId  : num  231 480 586 151 858 ...
##  $ rating   : num  5 5 5 3 2 3
##  $ timestamp: int  838983392 838983653 838984068 868246450 868245645 868245920
##  $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy (1995)
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Drama|Roman
##  - attr(*, ".internal.selfref")=<externalptr>
```

First six rows of the edx dataset for analysis and training of algorithms to generate predictions:

```
##   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
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 3  Action|Drama|Sci-Fi|Thriller
## 4       Action|Adventure|Sci-Fi
## 5 Action|Adventure|Drama|Sci-Fi
## 6       Children|Comedy|Fantasy
```

First six rows of the validation dataset for final RMSE evaluation:

```
##   userId movieId rating timestamp
## 1      1     231      5 838983392
## 2      1     480      5 838983653
## 3      1     586      5 838984068
## 4      2     151      3 868246450
## 5      2     858      2 868245645
## 6      2    1544      3 868245920
##                                                  title
## 1                                    Dumb & Dumber (1994)
## 2                                    Jurassic Park (1993)
## 3                                       Home Alone (1990)
## 4                                          Rob Roy (1995)
## 5                                     Godfather, The (1972)
## 6 Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
##                                    genres
## 1                                  Comedy
## 2         Action|Adventure|Sci-Fi|Thriller
## 3                           Children|Comedy
## 4                 Action|Drama|Romance|War
## 5                              Crime|Drama
## 6 Action|Adventure|Horror|Sci-Fi|Thriller
```

Rating being the variable of interest for which the predictions are to be generated, the other data present in the dataset was examined in terms of its distribution for having potential contribution in the variation in rating.

Distribution of the count of ratings received by each movie, and the count of ratings provided by each user were examined directly by grouping the data based on the *userId* and *movieId* variable. The *timestamp* variable was converted to *rating_year* by means of the *lubridate* package. Additionally, every entry in the *title* column has the movie title along with the year of it's release written in brackets after it. This regular pattern of entries was put to use for string processing using *regex* (regular expressions). Finally, to quantify the effect based on genres, the following method was used:

1. Each movie was observed to have either "(no genres listed)", or one or more genre entered in the *genres* column. Movies having more than one genre have all of them entered in the genre column separated using the vertical bar sign, "|", as a delimiter.
2. A character vector was made by grouping entries in the *genres* column.
3. The *unnest_tokens* command was used to separate individual genre names, applying necessary corrections wherever required, such as in cases of hyphenated genre names. Thus, individual genres occuring across the entire dataframe were put as single entries in one character vector.
4. An arbitrary score was assigned to each individual category based on the order of occurrence in the created character vector, thus yielding the following dataframe:

```
##                genres score
## 1  (no genres listed)     1
## 2              Action     2
## 3           Adventure     3
## 4           Animation     4
## 5            Children     5
## 6              Comedy     6
## 7             Fantasy     7
## 8                IMAX     8
## 9              Sci-Fi     9
## 10              Drama    10
## 11             Horror    11
## 12            Mystery    12
## 13            Romance    13
## 14           Thriller    14
## 15              Crime    15
## 16                War    16
## 17            Western    17
## 18            Musical    18
## 19        Documentary    19
## 20          Film-Noir    20
```

5. This score, while being nominal, served as a quantitative representation of the occurrence of a specific genre in the entries of the *genres* variable.
6. Adding up individual scores for a particular entry in the *genres* column gave a total genre score which was a quantitative representation of all the genres that a particular movie belonged to. This was calculated using the following function:

Function to define genre scores:

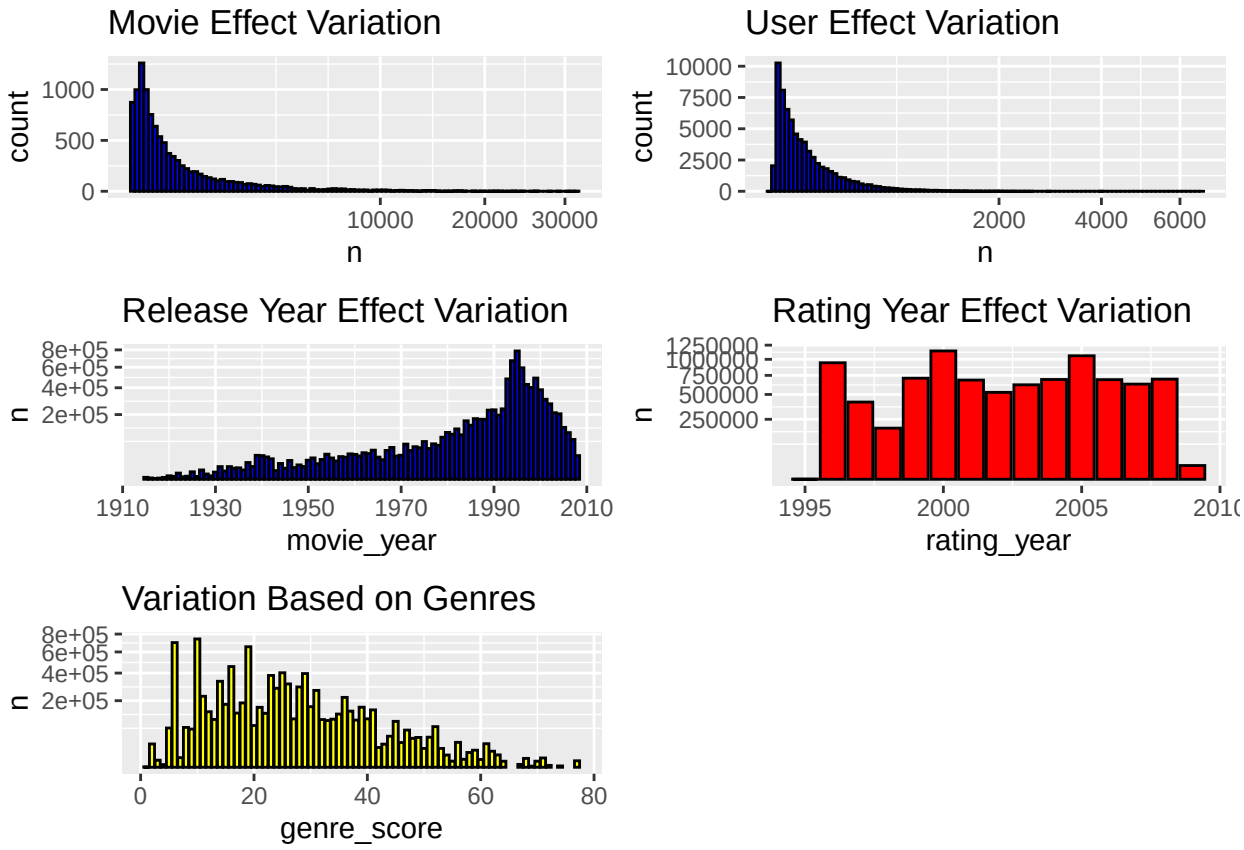
```
genre_score_fun <- function(g)
{g_dat <- g %>% str_split(pattern = "\|") %>% .[[1]] %>% data.frame(genre = .)
genre_score <- sapply(g_dat, function(genre){
ind <- match(genre, genre_ score_ chart$ genres)
genre_score_chart$score[ind]})
sum(genre_score)}
```

Distributions of selected variables:

The variables available in the original dataset, and those obtained by the wrangling and processing of data as described were examined for their distribution in the dataset to check if they have any kind of a skewed distribution, thus, making them potential sources of variability in rating prediction.

The variables considered were:
* *movieId*
* *userId*
* *rating_year*
* *movie_year* (movie release year)
* *genre_score*

Variability in rating observed based on selected potential sources of variation:



The graphs obtained show that the variables *movieId*, *userId* and *movie_year* are significantly skewed in their distribution, thus indicating a strong effect on the number of ratings. The *genre_score* variable shows some variability in it's effect on the number of ratings, while the *rating_year* variable does not show any significant variability.

Hence, *movieId*, *userId*, *movie_year*, and *genre_score* were the variables chosen for further analysis.

The linear model, ***y = mu + b_i + b_u + b_y + b_g + epsilon***, was applied using an averaging based approach. This was done as follows:

- The average of all ratings in the dataset was calculated mu = 3.51
- The movie bias is then calculated as: b_i = mean(rating - mu)
- The predictions using movie bias are then calculated as: pred_b_i = mu + b_i
- This is repeated for the remaining variables as follows:
  - b_u = mean(rating - mu - b_i)
  - pred_b_u = mu + b_i + b_u
  - b_y = mean(rating - mu - b_i - b_u)
  - pred_b_y = mu + b_i + b_u + b_y
  - b_g = mean(rating - mu - b_i - b_u - b_y)
  - pred_b_g = mu + b_i + b_u + b_y + b_g

Thus, we now construct a dataset which has the variables under consideration for being sources of variation in rating prediction, the effects of the biases induced by each of them, and the predictions generated by applying each individual bias to the average rating.

Dataset with variables chosen as sources of variation in rating prediction:

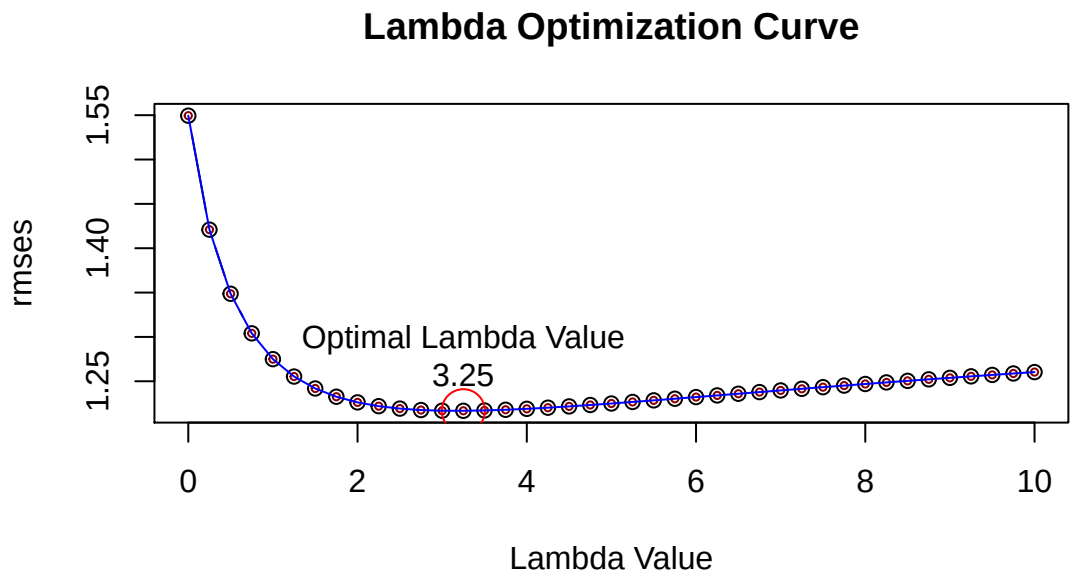
```
## Classes 'data.table' and 'data.frame':   6 obs. of  13 variables:
##  $ rating     : num  5 5 5 5 5 5
##  $ movieId    : num  122 185 292 316 329 355
##  $ userId     : num  1 1 1 1 1 1
##  $ movie_year : num  1992 1995 1995 1994 1994 ...
##  $ genre_score: num  19 31 35 14 24 18
##  $ b_i        : num  -0.6539 -0.3831 -0.0945 -0.1628 -0.175 ...
##  $ pred_b_i   : num  2.86 3.13 3.42 3.35 3.34 ...
##  $ b_u        : num  1.68 1.68 1.68 1.68 1.68 ...
##  $ pred_b_u   : num  4.54 4.81 5.1 5.03 5.02 ...
##  $ b_y        : num  0.00698 -0.03792 -0.03792 -0.032 -0.032 ...
##  $ pred_b_y   : num  4.54 4.77 5.06 5 4.98 ...
##  $ b_g        : num  -0.0054 -0.00312 0.00151 -0.01969 -0.00395 ...
##  $ pred_b_g   : num  4.54 4.77 5.06 4.98 4.98 ...
##  - attr(*, ".internal.selfref")=<externalptr>

##   rating movieId userId movie_year genre_score         b_i pred_b_i      b_u
## 1      5     122      1       1992          19 -0.65387934 2.858586 1.679235
## 2      5     185      1       1995          31 -0.38313118 3.129334 1.679235
## 3      5     292      1       1995          35 -0.09445454 3.418011 1.679235
## 4      5     316      1       1994          14 -0.16278816 3.349677 1.679235
## 5      5     329      1       1994          24 -0.17500816 3.337457 1.679235
## 6      5     355      1       1994          18 -1.02467799 2.487787 1.679235
##   pred_b_u          b_y pred_b_y          b_g pred_b_g
## 1 4.537821  0.006983816 4.544804 -0.005395510 4.539409
## 2 4.808569 -0.037918647 4.770650 -0.003120829 4.767529
## 3 5.097245 -0.037918647 5.059327  0.001505932 5.060833
## 4 5.028912 -0.032003181 4.996909 -0.019691473 4.977217
## 5 5.016692 -0.032003181 4.984689 -0.003945963 4.980743
## 6 4.167022 -0.032003181 4.135019 -0.014654944 4.120364
```

Lambda Optimization:

The effects of the biases and the predictions generated by them as shown above can be subject to regularization for better accuracy of prediction using a regularization parameter, *lambda*. For calculating the optimal value of *lambda*, the following procedure was used:

- A representative sample of 10000 ratings was chosen such that there were 1000 movies for each unique rating.
- This was divided into a training and test set.
- A range of *lambdas* from 0 to 10 with an interval of 0.25 was created as a numeric vector for each value to be tested individually for the RMSE value that it would yield.
- For n() being the number of ratings and the letter "l" being the value of *lambda*, the subsequent steps were followed.
- The average of all ratings in the dataset was calculated mu = 3.51
- The regularized movie bias is then calculated as: b_i_l = sum(rating - mu)/(n() + l)
- The regularized predictions using movie bias are then calculated as: pred_b_i_l = mu + b_i_l
- This is repeated for the remaining variables as follows:
  - b_u_l = sum(rating - mu - b_i_l)/(n() + l)
  - pred_b_u_l = mu + b_i_l + b_u_l
  - b_y_l = sum(rating - mu - b_i_l - b_u_l)/(n() + l)
  - pred_b_y_l = mu + b_i_l + b_u_l + b_y_l
  - b_g_l = sum(rating - mu - b_i_l - b_u_l - b_y_l)/(n() + l)
  - pred_b_g_l = mu + b_i_l + b_u_l + b_y_l + b_g_l

The following graph was obtained showing the RMSE for different values of *lambda* across the selected range:



It is seen that the RMSE was minimum at *lambda* = 3.25. This was the value used to generate regularized effect bias and prediction data.

Regularized Prediction Data:

Using all the above results, the following dataset was generated:

```
## Classes 'data.table' and 'data.frame':   6 obs. of  21 variables:
##  $ rating     : num  5 5 5 5 5 5
##  $ movieId    : num  122 185 292 316 329 355
##  $ userId     : num  1 1 1 1 1 1
##  $ movie_year : num  1992 1995 1995 1994 1994 ...
##  $ genre_score: num  19 31 35 14 24 18
##  $ b_i        : num  -0.6539 -0.3831 -0.0945 -0.1628 -0.175 ...
##  $ pred_b_i   : num  2.86 3.13 3.42 3.35 3.34 ...
##  $ b_u        : num  1.68 1.68 1.68 1.68 1.68 ...
##  $ pred_b_u   : num  4.54 4.81 5.1 5.03 5.02 ...
##  $ b_y        : num  0.00698 -0.03792 -0.03792 -0.032 -0.032 ...
##  $ pred_b_y   : num  4.54 4.77 5.06 5 4.98 ...
##  $ b_g        : num  -0.0054 -0.00312 0.00151 -0.01969 -0.00395 ...
##  $ pred_b_g   : num  4.54 4.77 5.06 4.98 4.98 ...
##  $ b_i_l      : num  -3.00e-04 -2.84e-05 -6.54e-06 -9.56e-06 -1.20e-05 ...
##  $ pred_b_i_l : num  3.51 3.51 3.51 3.51 3.51 ...
##  $ b_u_l      : num  0.0669 0.0669 0.0669 0.0669 0.0669 ...
##  $ pred_b_u_l : num  3.58 3.58 3.58 3.58 3.58 ...
##  $ b_y_l      : num  -3.43e-07 -8.94e-08 -8.94e-08 -6.12e-08 -6.12e-08 ...
##  $ pred_b_y_l : num  3.58 3.58 3.58 3.58 3.58 ...
##  $ b_g_l      : num  -3.48e-08 4.25e-07 -2.92e-07 -8.73e-08 -1.91e-07 ...
##  $ pred_b_g_l : num  3.58 3.58 3.58 3.58 3.58 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
##   rating movieId userId movie_year genre_score         b_i pred_b_i      b_u
## 1      5     122      1       1992          19 -0.65387934 2.858586 1.679235
## 2      5     185      1       1995          31 -0.38313118 3.129334 1.679235
## 3      5     292      1       1995          35 -0.09445454 3.418011 1.679235
## 4      5     316      1       1994          14 -0.16278816 3.349677 1.679235
## 5      5     329      1       1994          24 -0.17500816 3.337457 1.679235
## 6      5     355      1       1994          18 -1.02467799 2.487787 1.679235
##   pred_b_u          b_y pred_b_y          b_g pred_b_g         b_i_l pred_b_i_l
## 1 4.537821  0.006983816 4.544804 -0.005395510 4.539409 -2.997728e-04   3.512165
## 2 4.808569 -0.037918647 4.770650 -0.003120829 4.767529 -2.843854e-05   3.512437
## 3 5.097245 -0.037918647 5.059327  0.001505932 5.060833 -6.536533e-06   3.512459
## 4 5.028912 -0.032003181 4.996909 -0.019691473 4.977217 -9.557082e-06   3.512456
## 5 5.016692 -0.032003181 4.984689 -0.003945963 4.980743 -1.202537e-05   3.512453
## 6 4.167022 -0.032003181 4.135019 -0.014654944 4.120364 -2.119621e-04   3.512253
##        b_u_l pred_b_u_l         b_y_l pred_b_y_l         b_g_l pred_b_g_l
## 1 0.06685757   3.579023 -3.431769e-07   3.579023 -3.484433e-08   3.579023
## 2 0.06685757   3.579294 -8.942414e-08   3.579294  4.245029e-07   3.579295
## 3 0.06685757   3.579316 -8.942414e-08   3.579316 -2.917229e-07   3.579316
## 4 0.06685757   3.579313 -6.116911e-08   3.579313 -8.728339e-08   3.579313
## 5 0.06685757   3.579311 -6.116911e-08   3.579311 -1.910742e-07   3.579310
## 6 0.06685757   3.579111 -6.116911e-08   3.579111 -6.621431e-07   3.579110
```

Machine Learning Algorithm Training:

In the above dataset, the values for the variables *b_i*, *b_u*, *b_y*, *b_g*, *pred_b_i*, *pred_b_u*, *pred_b_y*, *pred_b_g*, *b_i_l*, *b_u_l*, *b_y_l*, *b_g_l*, *pred_b_i_l*, *pred_b_u_l*, *pred_b_y_l*, and *pred_b_g_l*, are all obtained by applying a set of uniform operations on the data previously present in the dataset. These variables would hence serve as a good predictor space for rating predictions, and thus were used as such.

Sample set creation:

With running machine learning algorithms on the full dataset being beyond the limitations of computing time and computing capacity, a random sampling approach was used. Random samples of 1000 movies were chosen for one run of each set of machine learning algorithms. Six such samples were taken by setting the seed at different points to obtain six different sample sets of 1000 movies each from the entire dataset. These were used to demonstrate consistency in reproducibility of results.

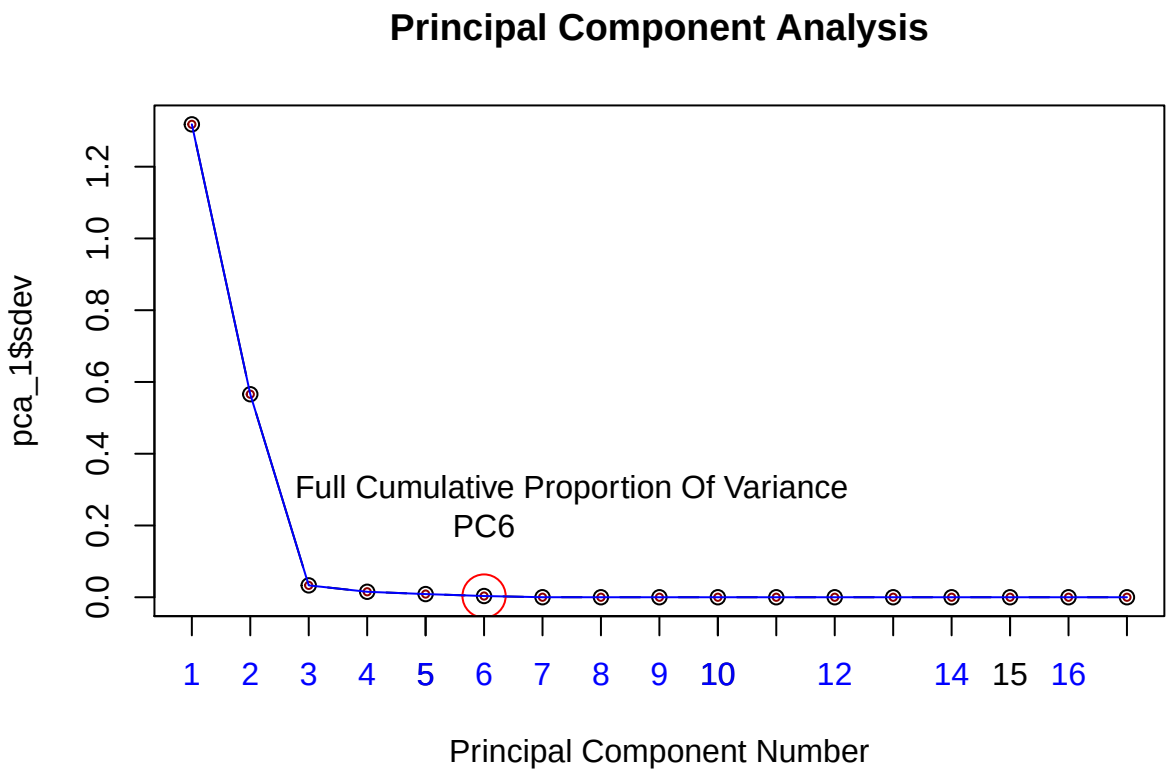
One particular sample set thus has a total of 1000 movies, with the average rating for each movie and the variables mentioned above as predictor space used for training the machine learning algorithms.

This sample set was then divided into a training set and test set comprising of 500 observations each.

The principal component analysis (for the first sample) showed the following results:

Principal Component Analysis of machine learning training data:

```
## Importance of components:
##                             PC1    PC2     PC3     PC4      PC5     PC6       PC7
## Standard deviation     1.3180 0.5658 0.03304 0.01524 0.008822 0.00340 2.374e-05
## Proportion of Variance 0.8438 0.1555 0.00053 0.00011 0.000040 0.00001 0.000e+00
## Cumulative Proportion  0.8438 0.9993 0.99984 0.99996 0.999990 1.00000 1.000e+00
##                               PC8       PC9      PC10      PC11      PC12
## Standard deviation     1.001e-05 1.041e-15 3.473e-16 1.812e-16 1.369e-16
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                              PC13      PC14      PC15      PC16      PC17
## Standard deviation     1.097e-16 9.599e-17 8.744e-17 6.466e-17 6.358e-17
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
```



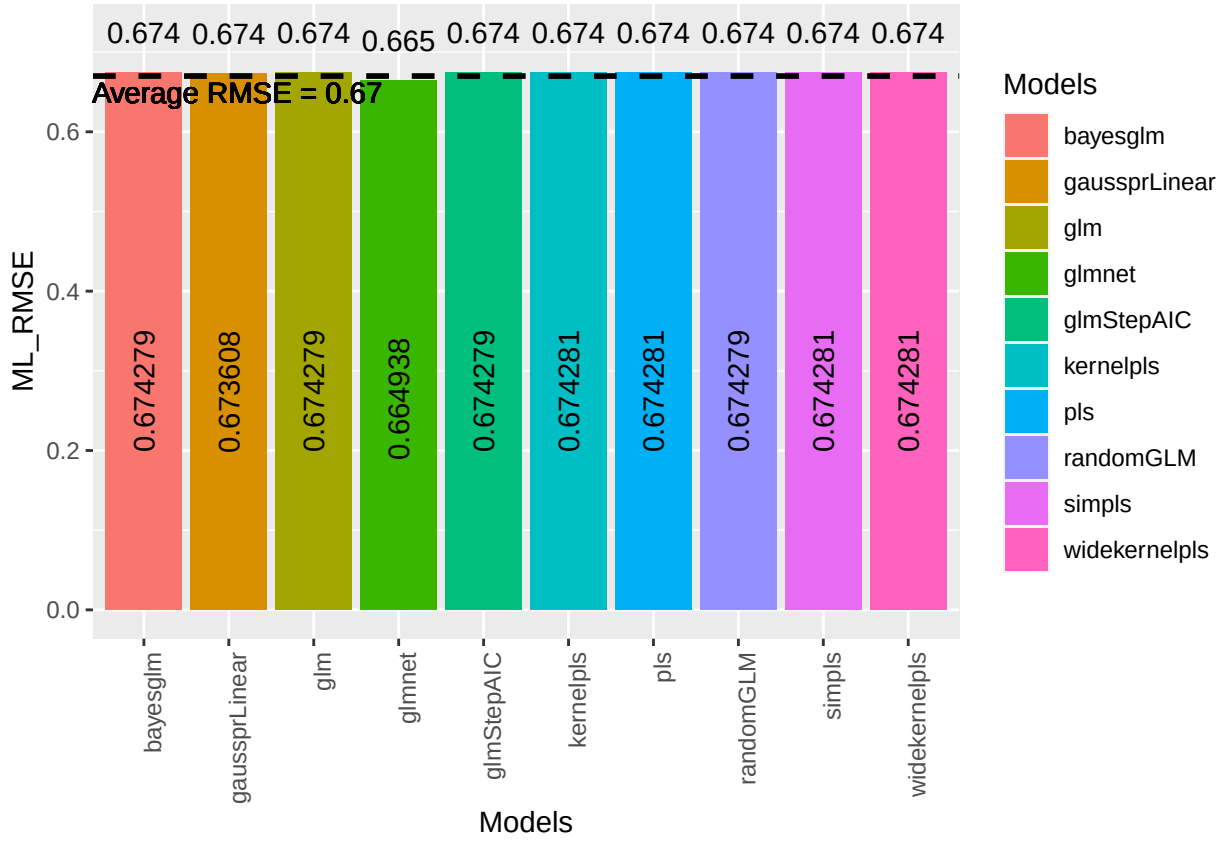
The data and the graph show that the first six principal components account for all the variability of the data. Thus, PC1 to PC6 were selected to train the machine learning algorithms.

Nine different linear classification models were chosen for machine learning training. These were “bayesglm”, “gaussprLinear”, “glm”, “glmStepAIC”, “glmnet”, “pls”, “simpls”, “kernelpls”, and “widekernelpls”. Additionally one ensemble model, “randomGLM”, was chosen. These models each had *ncomp* (number of components) as the only tuning parameter. With this parameter already being chosen optimally by principal component analysis, there was no further need for optimization, thus reducing computing effort, and making the choice of these models aptly suited for training and prediction purposes here.

Results:

The first run all all models yielded an average RMSE of 0.67 and a minimum RMSE of 0.66 as can be seen from following graph:

RMSE Graph:




```
#Minimum RMSE obtained:
min(final_rmses_1$ML_RMSE)
```

```
## [1] 0.6649379
```

```
#Model yielding the minimal RMSE:
min_ind <- which.min(final_rmses_1$ML_RMSE)
final_rmses_1[min_ind,]
```

```
##   Models   ML_RMSE rmse_rounded
## 5 glmnet 0.6649379        0.665
```

```
#Average RMSE from all models:
result <- mean(final_rmses_1$ML_RMSE)
result
```

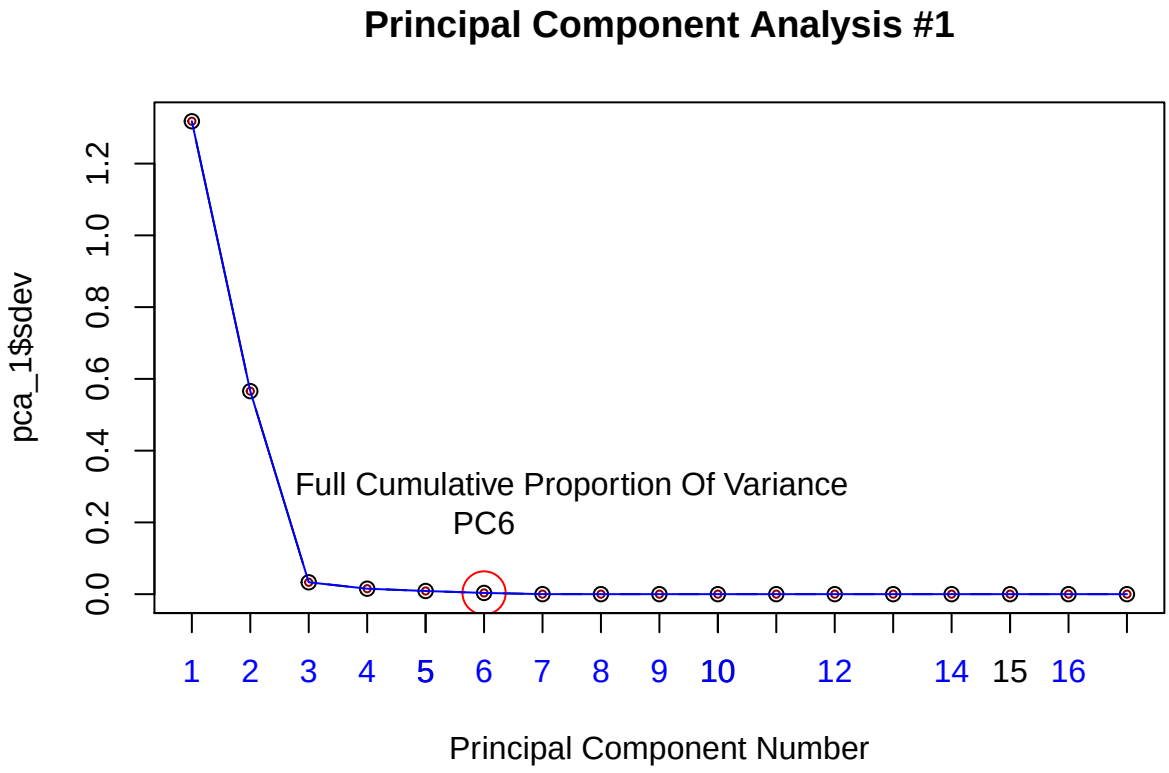
```
## [1] 0.6732785
```

Reproduced results:

To demonstrate consistency in reproducibility of results obtained by this method all models were run on six different samples. All samples consistently showed the first six components to account for all variability in the data as can be seen from the PCA data and graphs below:

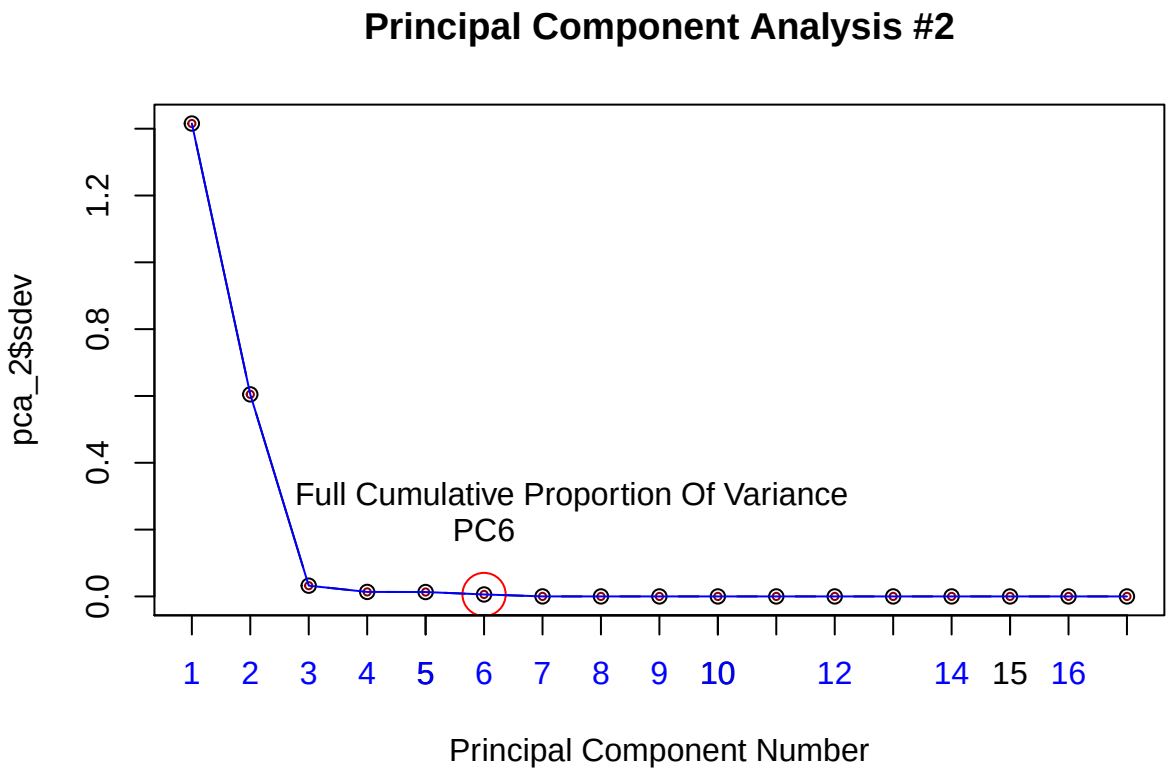
PCA_1:

```
## Importance of components:
##                             PC1    PC2     PC3     PC4      PC5     PC6       PC7
## Standard deviation     1.3180 0.5658 0.03304 0.01524 0.008822 0.00340 2.374e-05
## Proportion of Variance 0.8438 0.1555 0.00053 0.00011 0.000040 0.00001 0.000e+00
## Cumulative Proportion  0.8438 0.9993 0.99984 0.99996 0.999990 1.00000 1.000e+00
##                               PC8       PC9      PC10      PC11      PC12
## Standard deviation     1.001e-05 1.041e-15 3.473e-16 1.812e-16 1.369e-16
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                              PC13      PC14      PC15      PC16      PC17
## Standard deviation     1.097e-16 9.599e-17 8.744e-17 6.466e-17 6.358e-17
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
```



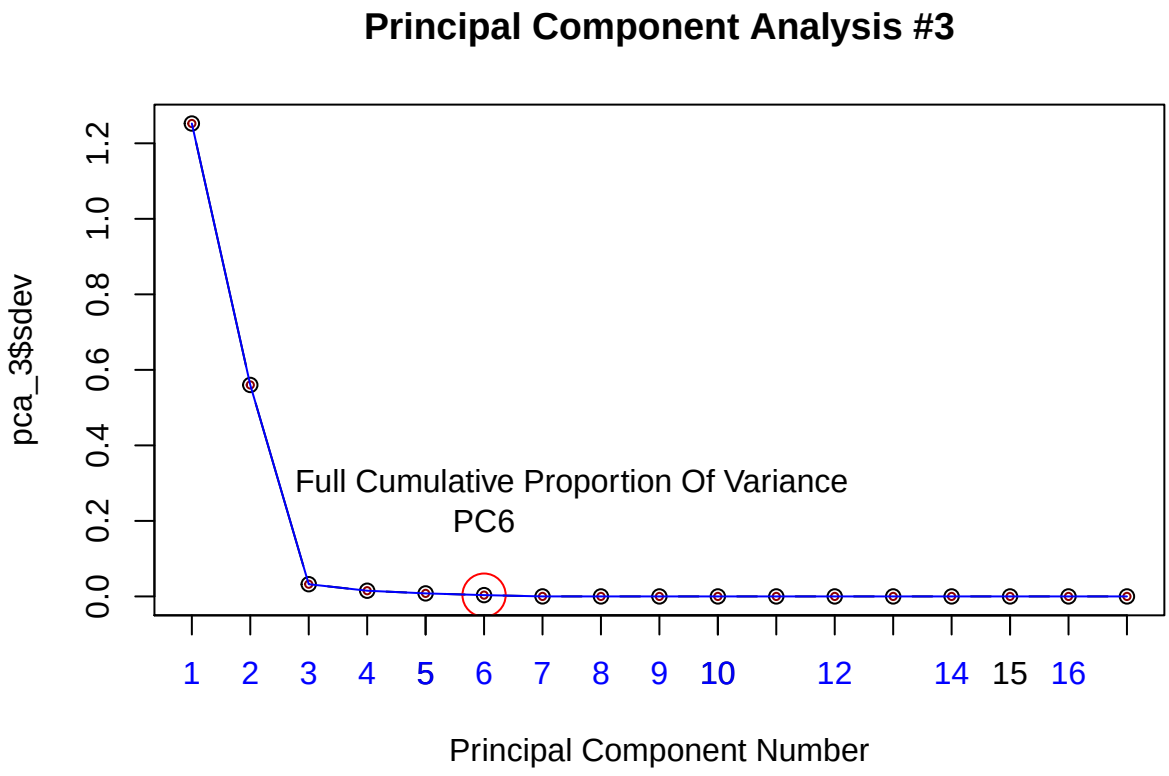
PCA_2:

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6       PC7
## Standard deviation     1.4155 0.6048 0.03218 0.01348 0.01315 0.00586 1.779e-05
## Proportion of Variance 0.8451 0.1543 0.00044 0.00008 0.00007 0.00001 0.000e+00
## Cumulative Proportion  0.8451 0.9994 0.99984 0.99991 0.99999 1.00000 1.000e+00
##                            PC8       PC9      PC10      PC11      PC12
## Standard deviation     2.616e-06 7.855e-16 3.842e-16 3.012e-16 1.252e-16
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                           PC13      PC14      PC15      PC16      PC17
## Standard deviation     1.164e-16 9.126e-17 8.019e-17 6.968e-17 6.647e-17
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
```



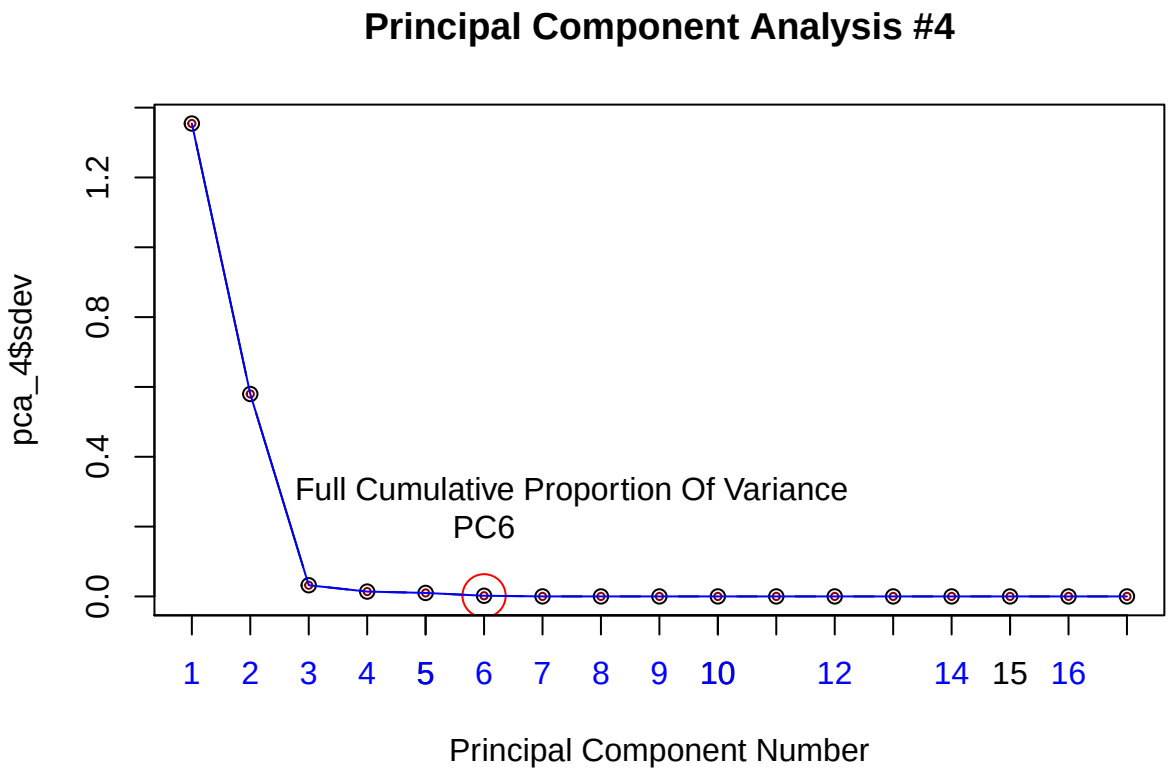
PCA_3:

```
## Importance of components:
##                             PC1    PC2     PC3     PC4      PC5      PC6
## Standard deviation     1.2524 0.5600 0.03234 0.01509 0.008014 0.003466
## Proportion of Variance 0.8328 0.1665 0.00056 0.00012 0.000030 0.000010
## Cumulative Proportion  0.8328 0.9993 0.99984 0.99996 0.999990 1.000000
##                               PC7       PC8       PC9      PC10      PC11
## Standard deviation     1.588e-05 1.121e-05 5.045e-16 4.614e-16 1.983e-16
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                              PC12      PC13      PC14      PC15    PC16      PC17
## Standard deviation     1.13e-16 9.809e-17 8.485e-17 7.634e-17 7.1e-17 5.955e-17
## Proportion of Variance 0.00e+00 0.000e+00 0.000e+00 0.000e+00 0.0e+00 0.000e+00
## Cumulative Proportion  1.00e+00 1.000e+00 1.000e+00 1.000e+00 1.0e+00 1.000e+00
```



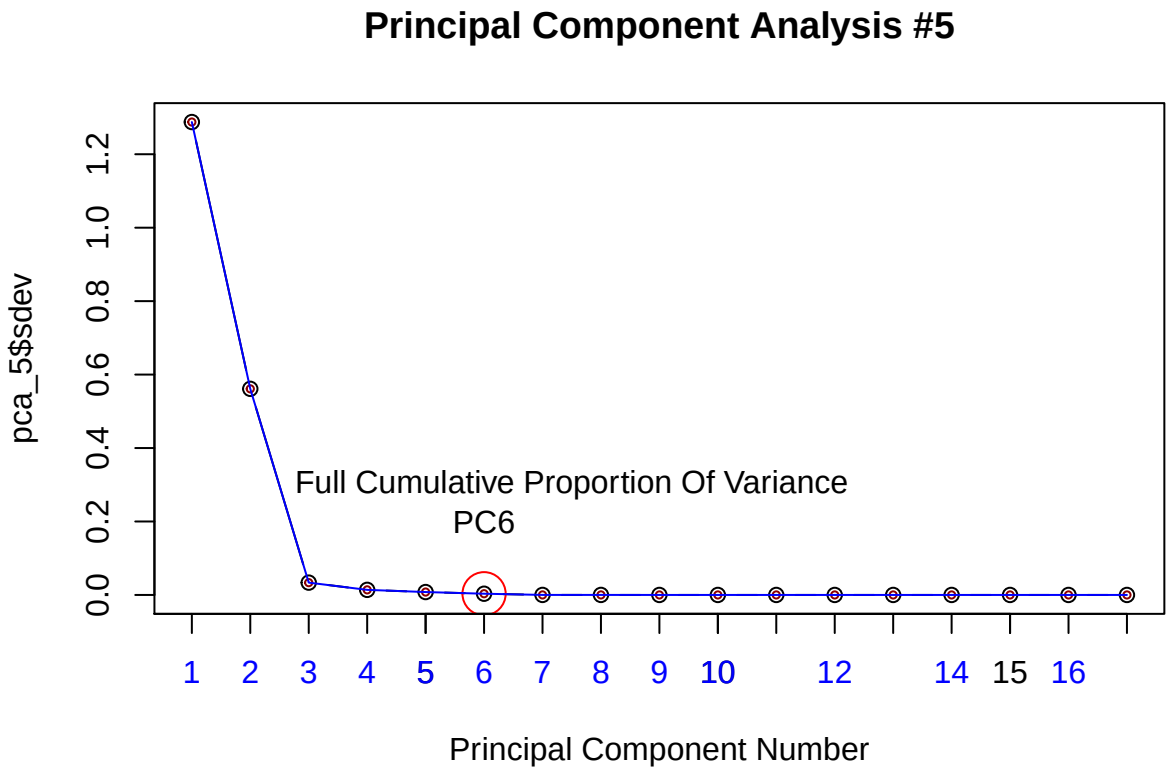
PCA_4:

```
## Importance of components:
##                             PC1    PC2     PC3     PC4     PC5      PC6       PC7
## Standard deviation     1.3545 0.5797 0.03198 0.01386 0.00996 0.001909 2.262e-05
## Proportion of Variance 0.8447 0.1547 0.00047 0.00009 0.00005 0.000000 0.000e+00
## Cumulative Proportion  0.8447 0.9994 0.99986 0.99995 1.00000 1.000000 1.000e+00
##                               PC8       PC9      PC10      PC11      PC12
## Standard deviation     7.996e-06 4.539e-16 3.448e-16 2.413e-16 1.597e-16
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                               PC13      PC14      PC15      PC16      PC17
## Standard deviation     1.227e-16 1.018e-16 8.801e-17 7.003e-17 6.225e-17
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
```



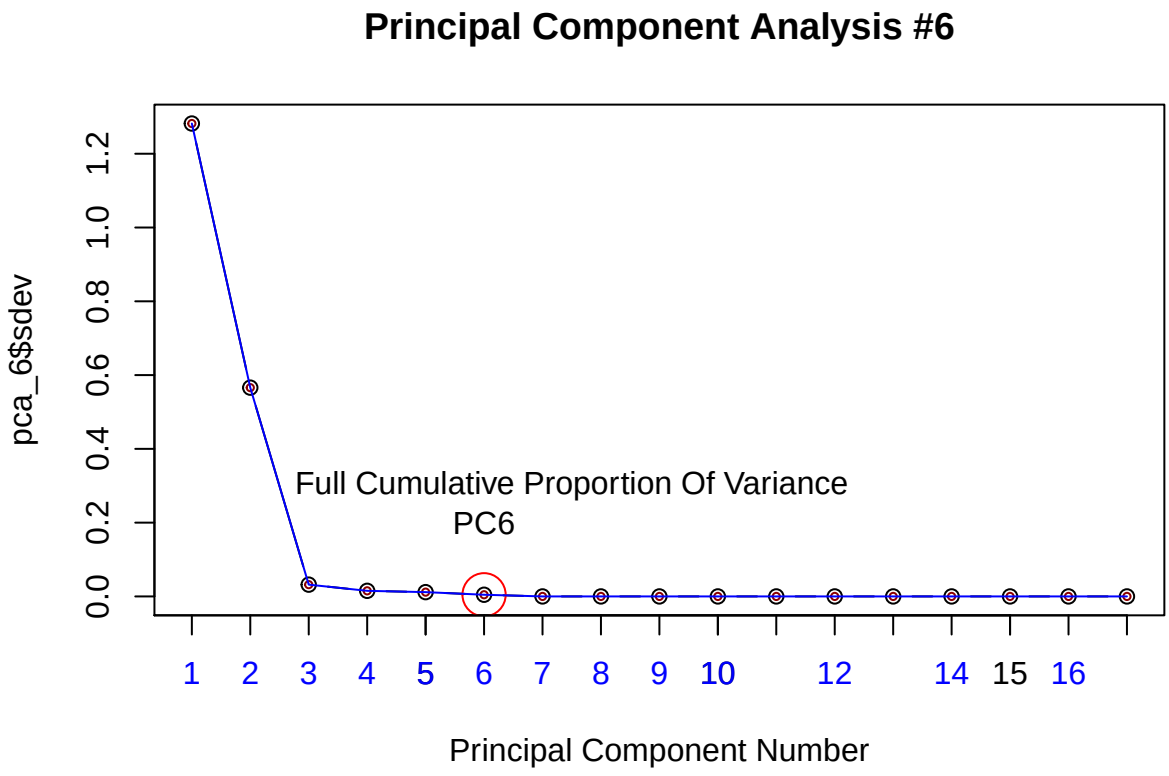
PCA_5:

```
## Importance of components:
##                             PC1    PC2     PC3     PC4      PC5      PC6
## Standard deviation     1.2876 0.5612 0.03354 0.01385 0.007882 0.003315
## Proportion of Variance 0.8398 0.1595 0.00057 0.00010 0.000030 0.000010
## Cumulative Proportion  0.8398 0.9993 0.99987 0.99996 0.999990 1.000000
##                               PC7       PC8       PC9      PC10    PC11
## Standard deviation     1.979e-05 1.215e-05 5.026e-16 2.612e-16 1.5e-16
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.0e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.0e+00
##                              PC12      PC13     PC14      PC15     PC16
## Standard deviation     1.187e-16 1.122e-16 1.01e-16 8.038e-17 6.82e-17
## Proportion of Variance 0.000e+00 0.000e+00 0.00e+00 0.000e+00 0.00e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.00e+00 1.000e+00 1.00e+00
##                              PC17
## Standard deviation     5.708e-17
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```



PCA_6:

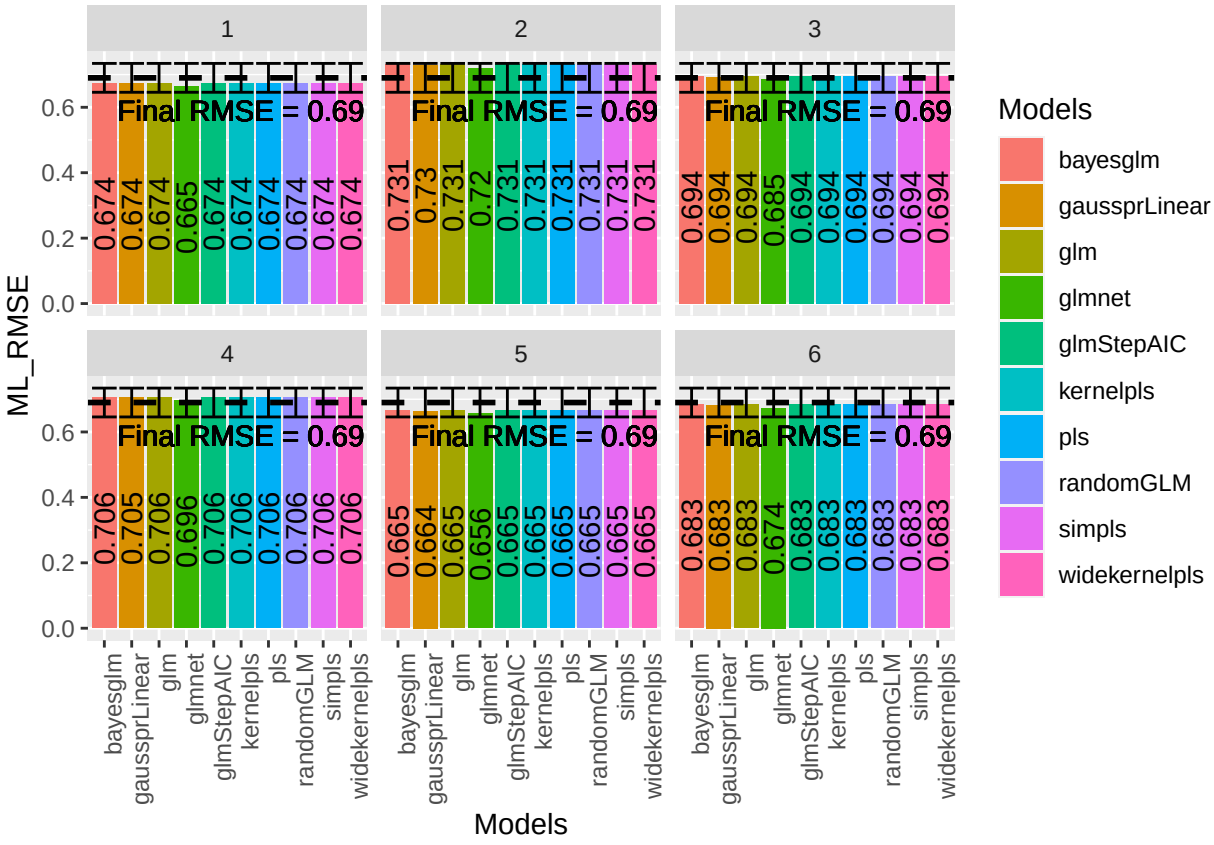
```
## Importance of components:
##                             PC1    PC2     PC3     PC4     PC5      PC6       PC7
## Standard deviation     1.2819 0.5659 0.03194 0.01519 0.01160 0.004495 7.941e-06
## Proportion of Variance 0.8363 0.1630 0.00052 0.00012 0.00007 0.000010 0.000e+00
## Cumulative Proportion  0.8363 0.9993 0.99980 0.99992 0.99999 1.000000 1.000e+00
##                               PC8       PC9      PC10      PC11      PC12
## Standard deviation     4.737e-06 3.577e-16 2.651e-16 1.709e-16 1.327e-16
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                               PC13      PC14      PC15      PC16      PC17
## Standard deviation     1.152e-16 8.267e-17 7.534e-17 6.061e-17 5.622e-17
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
```



Final Results:

All runs of all models on each sample set show RMSEs being within two standard deviations of the average RMSE, 0.69, and a standard deviation as low as 0.022, thus indicating good accuracy of models and consistent reproducibility of results, as can be seen from the graph below:

Graph showing results of all sample analyses combined:




```
# Minimum RMSE obtained:
min(f_rmse_chart$ML_RMSE)
```

```
## [1] 0.6558688
```

```
# Model yielding the minimal RMSE:
min_ind <- which.min(f_rmse_chart$ML_RMSE)
f_rmse_chart[min_ind,]
```

```
##    Models   ML_RMSE rmse_rounded sample_num
## 45 glmnet 0.6558688        0.656          5
```

```
# Standard Deviation of RMSEs from all models on all samples:
sd_rmse
```

```
## [1] 0.02206774
```

```r
# Average RMSE from all models:
final_result <- mean(f_rmse_chart$ML_RMSE)
final_result
```

```
## [1] 0.691203
```

Conclusion:

Among the variables present in the provided dataset, the *movieId* and *userId* variables were clear sources of variability in ratings of movies. The genre(s) and release year of the movies contributed to additional variability. With these variables in consideration, the linear model as described previously was fit to the data:

$$y = mu + b\_i + b\_u + b\_y + b\_g + epsilon$$

where,
y = **predicted rating,**
mu = **average rating,**
b_i = **movie bias,**
b_u = **user bias,**
b_y = **movie release year bias,**
b_g = **genre bias,**
$epsilon$ = **independent errors from sampling from the same dataset.**

The various "biases" and the predictions obtained by applying them across the dataset, along with their equivalents regularized with the optimized regularization parameter, *lambda* = 3.25, served as a proper predictor space for rating prediction. Principal component analysis of this predictor space showed the first six principal components to account for all variability in the data, hence these six principal components were chosen to train machine learning algorithms on. Nine individual machine learning models were chosen in addition to one ensemble model, and RMSEs obtained by each of these were averaged to yield the final result of one sample run. These models were chosen considering that *ncomp* (number of components) was the only tuning parameter for them. This parameter was already optimized by principal component analysis, consistently showing the first six principal components to account for all variability in each different sample set, and hence required no further optimization. This reduced computing time and effort.

The results were reproduced six times with different samples, yielding an average **final RMSE** of **0.69**, and showing a **standard deviation** as low as **0.022**, thus indicating good reproducibility and accuracy of this method.

The limitations of this analysis mainly comprised of limited sample sizes (10000 for *lambda* optimization and 1000 for each set of machine learning models). While the method's consistent reproducibility was demonstrated by repeated random sampling and a low standard deviation (0.022) in the results obtained, there is no guaranteed indicator of scalability for larger datasets. Limited computing capacity imposed these restrictions on sample sizes and computing time. Future goals would involve testing this method on larger datasets, but this can only be done with an increase in computing capacity beyond what is available with regular commercial laptops, and would require more powerful workstations.