HarvardX Data Science Capstone: MovieLens

Michael Lewis

09 January 2020

Executive Summary

Inspired by the Netflix Challenge, the goal of this project was to build a predictive model for movie ratings based on a dataset provided by GroupLens - a research lab from the University of Minnesoa. The data set includes $10M^+$ movie ratings, spanning more than 60,000 users and 10,000 movies.

Initial data handling and partitioning were performed per HarvardX PH125.9 course instructions. Subsequently, the data were checked for missing values to ensure integrity, a test set was generated, and exploratory data analyses were performed. Data visualization and feature engineering guided linear model development. Four models were evaluated, with final model accounting for movie, user, and temporal effects. This model produced was reasonably predictive, producing an RMSE of 0.866 when evaluated on new (validation) data.

Methods & Analysis

Exploratory Data Analysis

Prior to performing any statistical analyses or model building, several exploratory steps were undertaken. First an intermediate sample was drawn (n = 10,000) to expedite the data visualizations. From here, a distribution of the movie ratings was generated showing a slight negative skew (see **Figure1**). Next, the number of unique users and movies in the 'edX' set were determined (see **Table 1**). Checks for missing values on these two fields was performed and return no results.

```r
# Create small set with which to generate charts quickly
set.seed(1, sample.kind="Rounding")  # If using R <v3.6, use set.seed(1)
edaTbl <- sample_n(edx, 10000)
```

Figure 1. Distribution of Movie Ratings

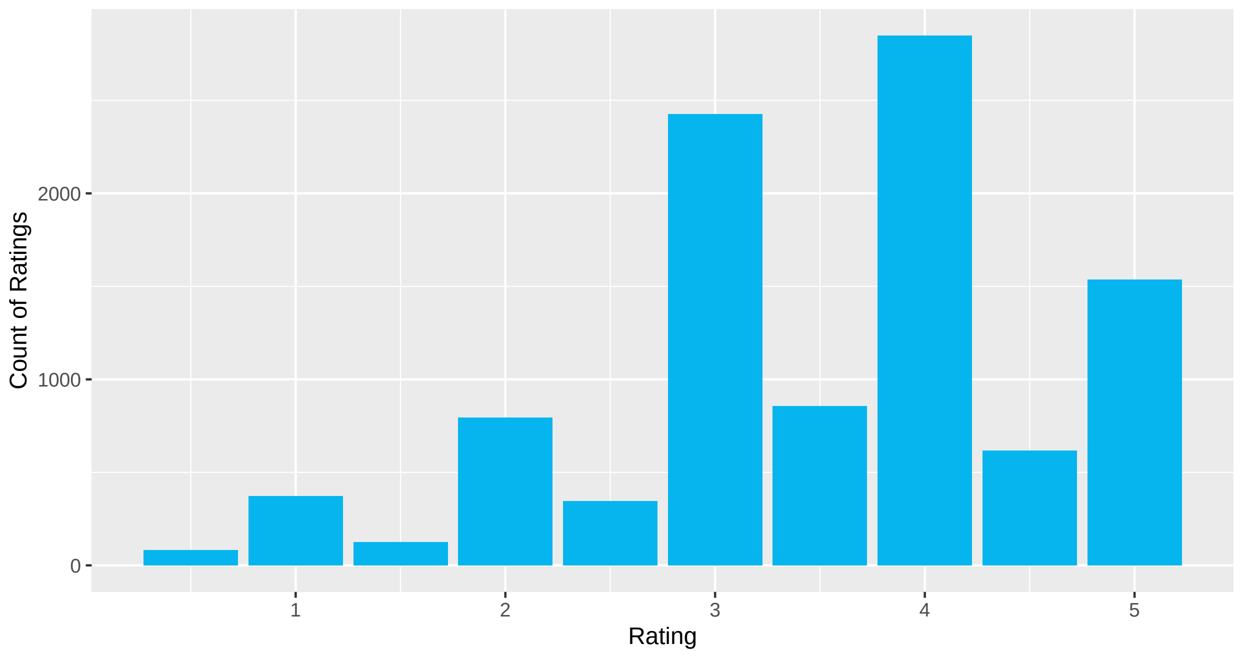
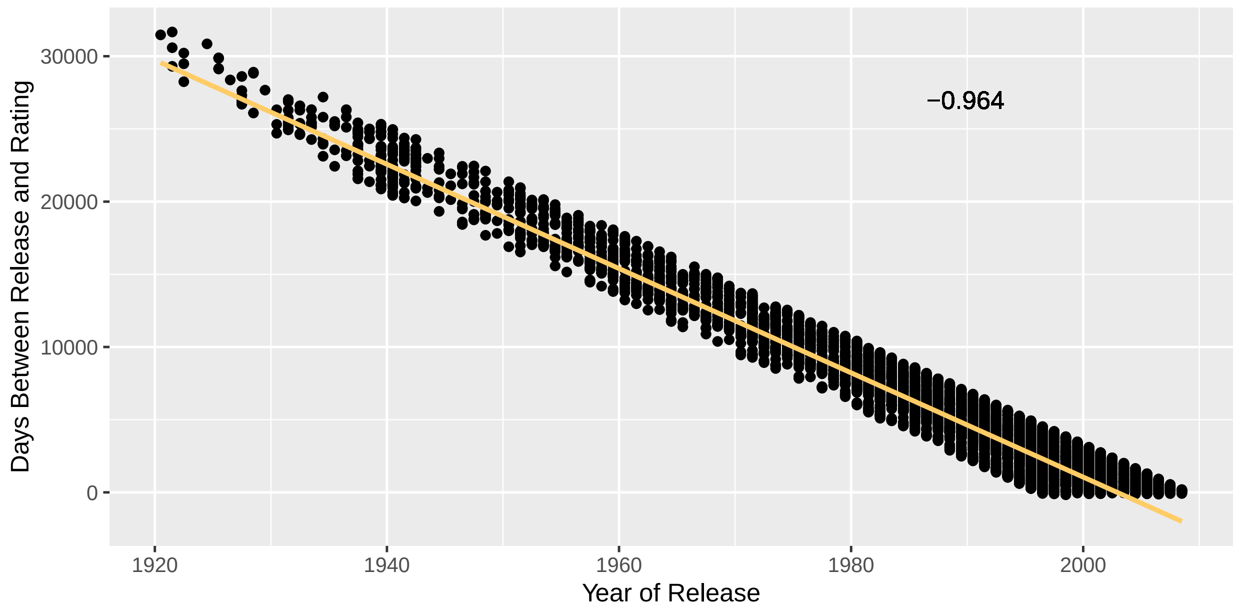


Table 1: Counts of Unique Users and Movies

Unique Users	Unique Movies
69878	10677

Next, the ‘timestamp’ column was converted to a date field, and a ‘release year’ field was extracted from the movie ‘title’ (assuming a mid-year release). From here, a ‘ratings lag’ field was calculated by subtracting the ‘release year’ from the date ‘timestamp’ field. In order to assess whether the an overall rating trend over time was distinct from looking back fondly on older movies, **Figure 2** was produced. Given the high correlation between the age of the movie and the lag-to-rating, no such distinction was observed. This inability to disentangle a hindsight bias from a general temporal trend is driven largely by the fact that movie ratings began in 1995, eighty years after the release of the earliest movie in the data set. As such, the ‘release year’ field was dropped prior to further analysis.

Figure 2. Movie Age vs. Rating Lag

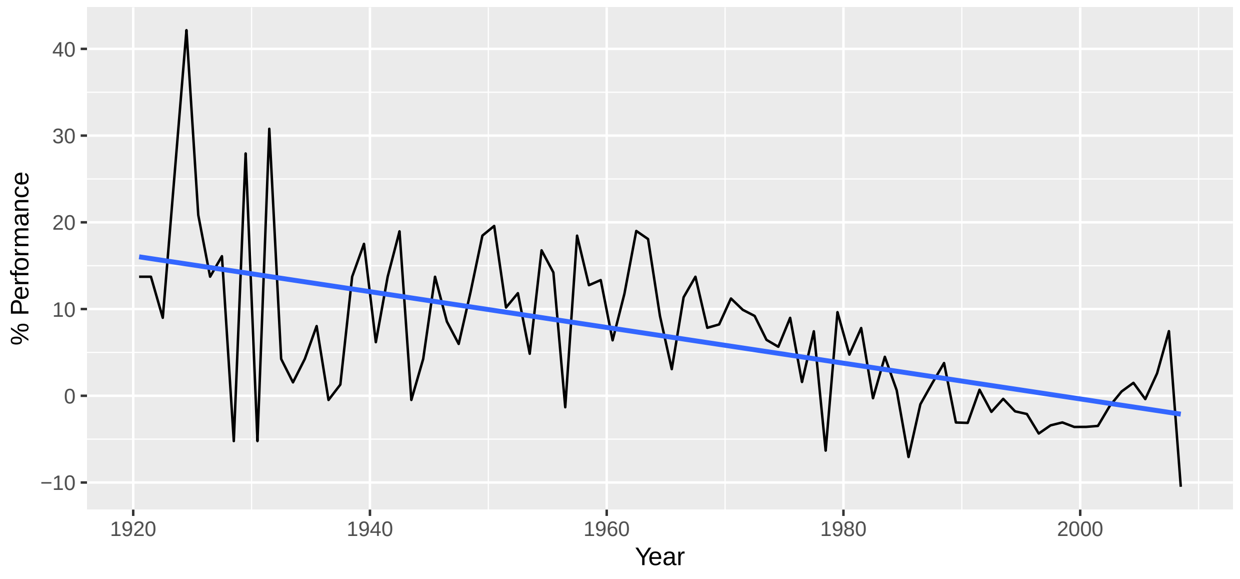


To facilitate an assessment of movie ratings over time, a relative performance measure was derived by calculating the percent deviation from the overall rating average ($\overline{x}$) for all the movies in a given year ($\hat{x}_i$). This measure underlies the temporal analysis shown in **Figure 3**.

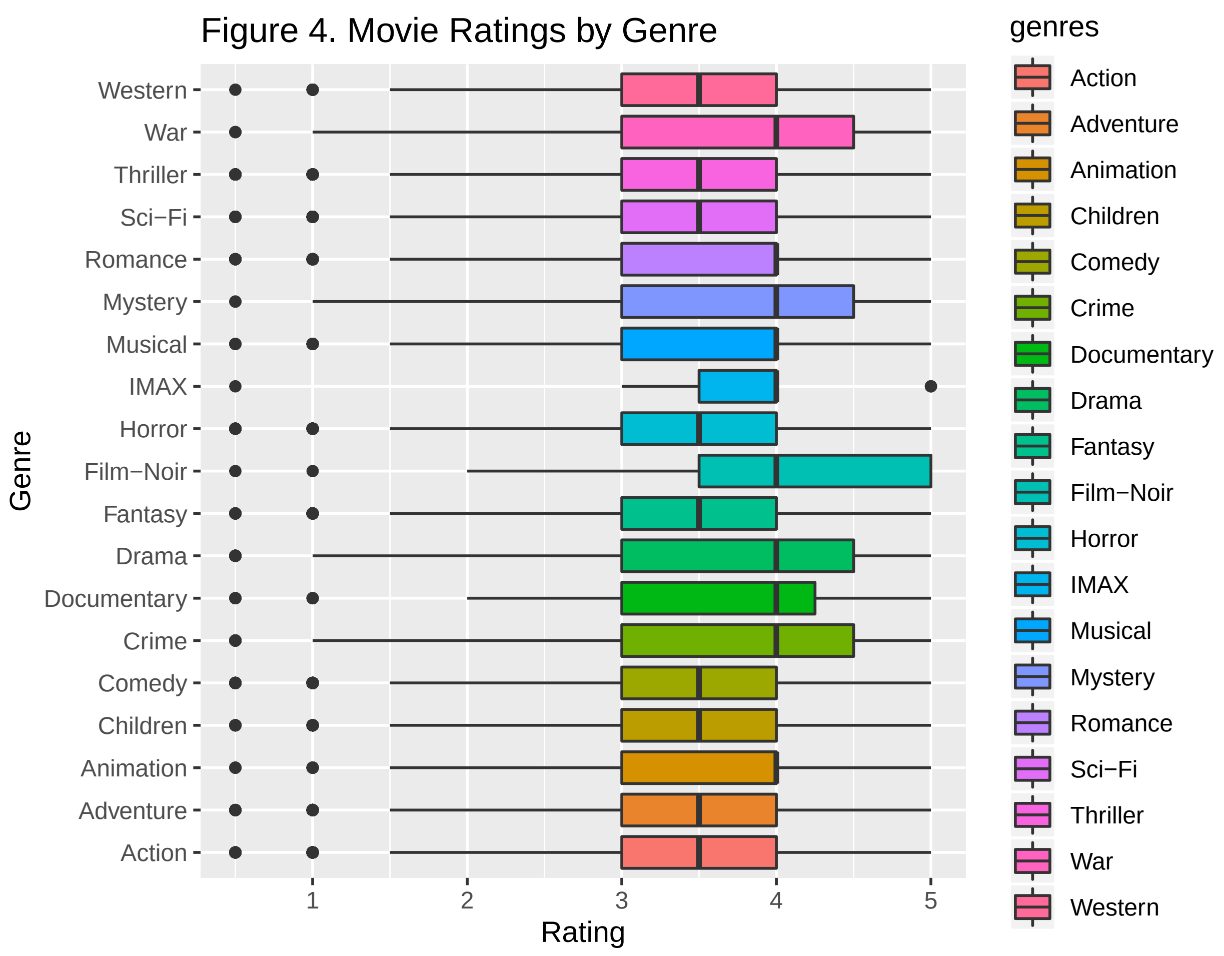
$$RelativeAnnualPerformance = \frac{\hat{x}_i - \overline{x}}{\overline{x}} * 100$$

Figure 3. Time Series Analysis of Ratings

YOY Relative Ratings Performance



Next, to explore potential genre-level differences in ratings, the delimited values in the 'genres' field were parsed to produce **Figure 4** . The rating distributions for certain genres are similar and indeed likely to co-occur while, others were distinct. While genre was not ultimately incorporated in the models, future analyses could leverage this field to improve predictions (more on this point in the 'Conclusion' section).



A final step prior to model building was to split the 'edX' set was split into training (90%) and testing (10%) sets.

```
# Now let's perform some analytics
  # Subset dataset into training and testing sets
  set.seed(1, sample.kind="Rounding") # If using R <v3.6, use set.seed(1)
  test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
  training <- edx[-test_index,]
  temp <- edx[test_index,]

  # Check to ensure all 'users' and 'movies' from test are in training set
  testing <- temp %>%
    semi_join(training, by = "movieId") %>%
    semi_join(training, by = "userId")

    # Add back missing rows to training set and remove extraneous data
    withheld <- anti_join(temp, testing)
    training <- rbind(training, withheld)
```

Model Evaluation

The model evaluation criterion used is the root mean squared error (RMSE), which captures the deviation of the predicted values (p_i) from the actual values (a_i) as shown below.

$RMSE = \sqrt{\sum_{i=1}^{n}(a_i - p_i)^2/n}$

Results

The first model trained used the average movie rating as the prediction for all values. This naive model served as a baseline for future models, with an RMSE of 1.06. Subsequent models leveraged the detrending of grouped means. This approach allows for the incremental accounting of feature-specific bias in the model.

```r
# MODEL 0 - A Naive Model, predicitng average movie rating for all obs.
mu_0 <- mean(training$rating)

  # Evaluate MOdel 0 on 'test' data & add to table for model comparison
  naiveRMSE <- RMSE(testing$rating, mu_0)
  rmseResults <- tibble(Method = "Niave Model (average rating)", RMSE = naiveRMSE)
```

Model 1 accounted for movie-specific effects and reduced the RMSE to 0.943. Adding the rater-effects in Model 2 further reduced the RMSE to 0.865.

```r
# MODEL 1 - Linear regression controlling for 'movie' effects
m1 <- training %>%
  group_by(movieId) %>%
  summarize(b_mov = mean(rating - mu_0))

predictedRatings <- mu_0 + testing %>%
  left_join(m1, by = 'movieId') %>%
  .$b_mov

# Evaluation
  m1RMSE <- RMSE(predictedRatings, testing$rating)
  rmseResults <- bind_rows(rmseResults,
                           tibble(Method = "Movie Effects", RMSE = m1RMSE))
```

```r
# MODEL 2 - Model 1 + 'user' (rater) effects
m2 <- training %>%
  left_join(m1, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(b_user = mean(rating - mu_0 - b_mov))

predictedRatings <- testing %>%
  left_join(m1, by = 'movieId') %>%
  left_join(m2, by = 'userId') %>%
  mutate(pred = mu_0 + b_mov + b_user) %>%
  .$pred

# Evaluation
  m2RMSE <- RMSE(predictedRatings, testing$rating)
  rmseResults <- bind_rows(rmseResults,
                           tibble(Method = "Movie + User Effects", RMSE = m2RMSE))
```

Model 3 accounts for a general temporal rating trend an constitues an improvement of $3.54x10^{-4}$ in RMSE over model 2. **Table 2** shows the predictive performance for each model on the 'test' data. Given its superior performance, model 3 was selected for use on the 'validation' data.

```
# MODEL 3 - Model 2 + 'year' effects
m3 <- training %>%
  left_join(m1, by = 'movieId') %>%
  left_join(m2, by = 'userId') %>%
  group_by(releaseYear) %>%
  summarize(b_yr = mean(rating - mu_0 - b_mov - b_user))

predictedRatings <- testing %>%
  left_join(m1, by = 'movieId') %>%
  left_join(m2, by = 'userId') %>%
  left_join(m3, by = 'releaseYear') %>%
  mutate(pred = mu_0 + b_mov + b_user + b_yr) %>%
  .$pred

  # Evaluation
  m3RMSE <- RMSE(predictedRatings, testing$rating)
  rmseResults <- bind_rows(rmseResults,
                           tibble(Method = "Movie + User + Year Effects", RMSE = m3RMSE))
  rmseResults %>% knitr::kable(caption = "Comparison of Model Performance on Test Data")
```

Table 2: Comparison of Model Performance on Test Data

Method	RMSE
Niave Model (average rating)	1.0600537
Movie Effects	0.9429615
Movie + User Effects	0.8646843
Movie + User + Year Effects	0.8643301

The final result, from deployment of model 3 on the 'validation' data, is an RMSE of 0.8655043 (as seen in **Table 3**).

```
# Predicting on the 'Validation' data
predictedRatings <- validation %>%
  left_join(m1, by = 'movieId') %>%
  left_join(m2, by = 'userId') %>%
  left_join(m3, by = 'releaseYear') %>%
  mutate(pred = mu_0 + b_mov + b_user + b_yr) %>%
  .$pred

  # Evaluation
  validRMSE <- RMSE(predictedRatings, validation$rating)
  finalRes <- tibble(Method = "Model 3 - Validation Data", RMSE = validRMSE)
  finalRes %>% knitr::kable(caption = "Predictive Performance on Validation Data")
```

Table 3: Predictive Performance on Validation Data

Method	RMSE
Model 3 - Validation Data	0.8655043

```
# RMSE for predictions on validation data is 0.8655043.
```

Conclusion

This project's challenge was to produce a model that predicts movie ratings. The analytic approach undertaken here relied on initial exploratory analysis, data visualizations, and the construction of an intermediate feature to capture annualized movie performance. Iteratively, features were added and evaluated for their impact on the RMSE of predictions on an intermediate (test) data set. Evaluation against thes data, suggests the final model constitutes a greater than 18% improvement over the naive baseline. By accounting for movie and user biases as well as a general temporal trend, a final model was constructed that produced an **RMSE of 0.8655043** when applied to a new (validation) data.

While these results are promising, future work could likely improve predictive performance in the following ways:

(1) using cross-validation to increase the generalizabilty of the model generated,

(2) leveraging smoothing or polynomial techniques to model the temporal trend in movie ratings,

(3) performing a factor analysis on the genre tags and using these factors in the model, and

(4) accounting for an interaction (moderator) effect between users and genres - as individuals are likley to have differential genre-level predispositions about the quality of movies.

Lastly, it is important to note the boundary constraints associated with this analysis. If the model were to be applied to data from substantially different movies or to similar movies that were rated by a different set users (e.g. "expert" versus audience reviews), the *coefficients* generated here are likely to produce poor predictions. Thus, while the *analytical process* described above is transferrable to other applications of linear modeling, the specific models are less so.