Predicting Representation of Women in Films

Jae-Ho Lee, Dhruv Khanna, Emily Ciaccio

Abstract

Statistical learning methods were applied to movie data to predict the Bechdel test score based on pre-release movie data. A variety of learning techniques were explored and validated. Final results suggest that more data is needed to truly capture female influence in films. Further analysis that includes more comprehensive data on the movie is recommended.

Introduction

This analysis aims to predict the Bechdel test[1] score for a movie given a variety of features. The Bechdel test is a famous metric to gauge female representation in media. The requirement for any media to pass the test is for 2 named female characters to share dialogue about something other than a man. Through this analysis we will try to discover how factors like budget, cast and crew gender ratios, year of release, etc. influence a movie's Bechdel score.

The nature of the data used below allows us to apply a variety of statistical learning methods from the course. The response is categorical with a range of 0 - 3, which allows for both multiclass and binary classification tasks. The predictors include a mix of continuous and categorical variables; the structure of certain variables also allowed us to implement feature extraction techniques as seen below.

Methods

Data Description

The data used for this analysis is collected from two sources:

- A public API [2]
- Modified Kaggle version of the MovieLens Dataset [3]

[1]Wikipedia: Bechdel Test
[2]BechdelTest: Bechdel Test
[3]Kaggle: MovieLens Fataset

The first dataset contains the Bechdel test scores for over 8000 movies ranging from the early 1900s to 2019, thus providing us with a varied sample. The second dataset contains widespread information about movies thus allowing us to try a variety of modeling approaches. The two datasets are merged based on their IMDB IDs.

The specific task at hand here is classifying the multiclass variable `bechdel` and its binary counterpart `bechdel_bin`. Important predictors such as `female_ratio`, `female_director` as well as the various genre dummy variables are calculated from the `cast_crew` data set. Other predictors include budget and year of release.

We specifically focus on pre-release information to improve the overall applicability of this model. Some exploratory data analysis can be found in the appendix.

Modeling

In order to predict the Bechdel test score, a variety of binary and multiclassification techniques were used in addition to regression methods. Multiclassification tasks tried to predict a score between 0 - 3 whereas binary classification tasks simply predicted 0 (fail) or 1 (pass).

The modeling strategies considered were:

- KNN Classification (multi / binary)
- KNN Regression
- Stochastic Gradient Boosted (multi / binary classification)
- Multinomial Regression (multiclass)
- Logistic Regression (binary class)

Evaluation

The KNN regression model was selected based on RMSE from k = 1:100. All other models were tuned used 5-fold cross-validation through the `train` package. Multiclass models were tuned to maximize the default Accuracy, whereas binary models were tuned to maximize Area under the ROC curve.

Models were ultimately evaluated based on their ability to simply predict whether a movie passes or fails the Bechdel test (binary classification). The binary classification tasks was naturally more consistent due to the drop in factor levels, but the binary outcome improved class balance which ultimately helped fit better models. Confusion Matrices were used to evaluate performance on the training dataset in order to choose a final model.

Multiclass

```r
k = 1:100

knn_mods = map(k, ~ knnreg(bechdel ~ . - bechdel_bin, data = df_trn, k = .x))
knn_preds = map(knn_mods, ~ predict(.x, newdata = df_trn, type = "response"))
knn_rmse = map(knn_preds, ~ calc_rmse(act = df_trn$bechdel, pred = .x))

fit_knnreg = knnreg(bechdel ~ . - bechdel_bin, data = df_trn, k = which.min(knn_rmse))
```

```r
set.seed(42)
fit_knn_mult = train(factor(bechdel) ~ . - bechdel_bin - runtime, data = df_trn,
                method = "knn",
                trControl = trainControl(method = "cv", number = 5))
```

```r
set.seed(42)
fit_gbm_mult = train(factor(bechdel_bin) ~ . - bechdel - runtime, data = df_trn,
                     method = "gbm",
                     trControl = trainControl(method = "cv", number = 5),
                     verbose = FALSE)
```

```r
set.seed(42)
fit_multinom_mult = train(factor(bechdel) ~ . - bechdel_bin - runtime, data = df_trn,
                          method = "multinom",
                          trControl = trainControl(method = "cv", number = 5),
                          trace = FALSE)
```

Binary

```r
set.seed(42)
#response mutated using make.names() to allow custom metric
fit_knn_bin = train(make.names(factor(bechdel_bin)) ~ . - bechdel - runtime, data = df_trn,
                method = "knn",
                trControl = trainControl(method = "cv",
                                         number = 5,
                                         classProbs = TRUE,
                                         summaryFunction = twoClassSummary),
                metric = "ROC")
```

```r
set.seed(42)
fit_gbm_bin = train(make.names(factor(bechdel_bin)) ~ . - bechdel - runtime, data = df_trn,
                   method = "gbm",
                   trControl = trainControl(method = "cv",
                                            number = 5,
                                            classProbs = TRUE,
                                            summaryFunction = twoClassSummary),
                   verbose = FALSE, metric = "ROC")
```

```r
set.seed(42)
fit_logistic_bin = train(make.names(factor(bechdel_bin)) ~ . - bechdel - runtime, data = df_trn,
                         method = "glm",
                         trControl = trainControl(method = "cv",
                                                  number = 5,
                                                  classProbs = TRUE,
                                                  summaryFunction = twoClassSummary),
                         trace = FALSE, metric = "ROC")
```

Results

Ultimately, binary classification was the best option, both performance-wise and due to other reasons as stated above. The results show a similar performance across all models. KNN model is chosen as the final model due to slightly better performance across Accuracy, Sensitivity and Specificity.

Models	Accuracy	Sensitivity	Specificity
K-Nearest Neighbors	0.647	0.539	0.738
Stochastic Boosted Gradient	0.648	0.505	0.542
Logistic	0.646	0.542	0.734

Discussion

In order to truly evaluate the performance of this model, its predictive power is tested on a held-out testing dataset, the results of which can be seen below.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 140 162
##          1 209 251
##
##                Accuracy : 0.5131
##                  95% CI : (0.477, 0.5492)
##     No Information Rate : 0.542
##     P-Value [Acc > NIR] : 0.94893
##
##                   Kappa : 0.009
##
##  Mcnemar's Test P-Value : 0.01693
##
##             Sensitivity : 0.4011
##             Specificity : 0.6077
##          Pos Pred Value : 0.4636
##          Neg Pred Value : 0.5457
##              Prevalence : 0.4580
##          Detection Rate : 0.1837
##    Detection Prevalence : 0.3963
##       Balanced Accuracy : 0.5044
##
##        'Positive' Class : 0
##
```

We can see that the model fails to perform significantly on the held-out test data. The accuracy of `0.5381` is only slightly higher than the No Information Rate (accuracy if you blidnly guess the same thing for all observations) of `0.5354`.

There are a variety of reasons that could possibly explain these results. Based on the initial EDA, we can see that predictors like `female_ratio` and `female_director` don't drastically change the score distributions.

Table 1: Female Directors

	0	1
0	24	25
1	36	63

Table 2: Male Directors

	0	1
0	119	136
1	170	189

Subsetting the dataset into male and female directors shows that the model performs rather similarly for subsets, which further lends to the theory that these variables don't really contribute to the bechdel score on their own:

Female director subset:

Male director subset:

Perhaps the variables gathered here are simply not enough to detect a strong female influence in the creation of films. Model accuracy may improve with the involvement of more specific features such as:

- Ratio of male to female lead / supporting actors
- Female influence in the writer's room

It may also be naive to restrict analysis to pre-release movie data based on the production of the movie; perhaps an analysis that explores Natural Language Processing (NLP) to detect what types of dialogue characterize movies that pass the Bechdel test.

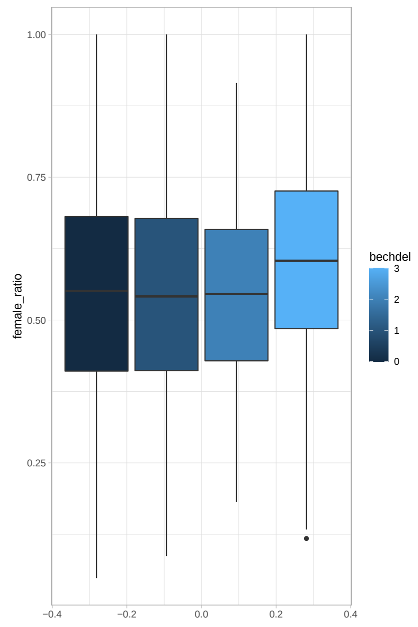
Appendix

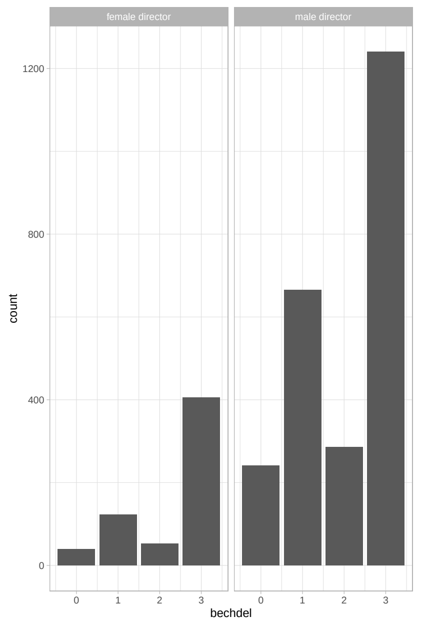
Data Dictionary

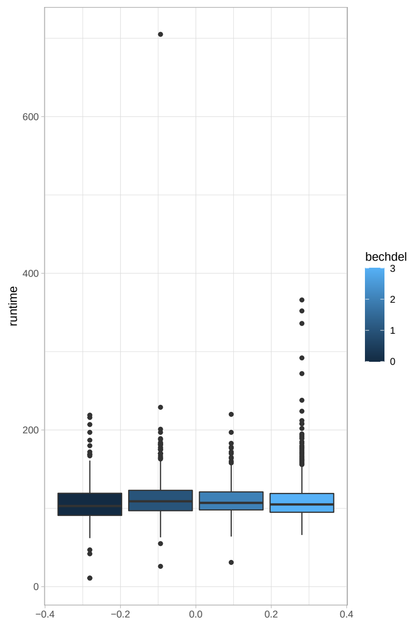
- `id` - MovieLens id that is used as a key between metadata dataset and cast-crew dataset
- `imdb_id` - IMDB id that is used as a key between metadata dataset and bechdel-score dataset
- `bechdel` - bechdel test score (0 - 3)
- `year` - year of movie release
- `belongs_to_collection` - dictionary item that holds collection name if true; N/A otherwise
- `budget` - movie budget ($)
- `genres` - dictionary holding genres for each movie
- `runtime` - total movie runtim (min)
- `title` - original movie title
- `female_director` - factor variable that is 1 when movie is directed by a female; 0 otherwise
- `female_ratio` - ratio of females to males in cast and crew for movie
- `bechdel_bin` - factor variable that is 1 when a movie passes the Bechdel test; 0 otherwise
- `action, adventure, animation, etc` - dummy variables for respective genres

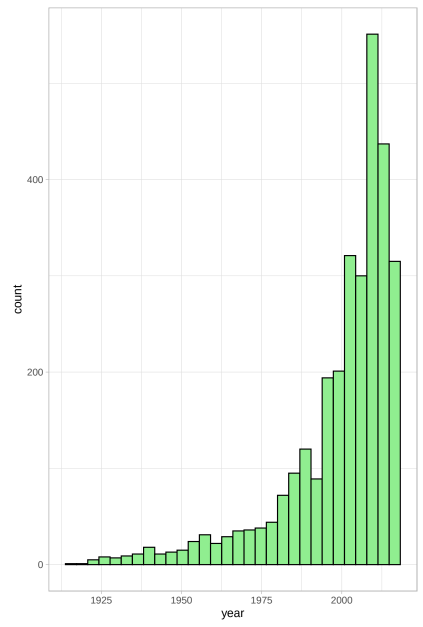
Additional documentation can be accessed from the links provided in the Data Description.

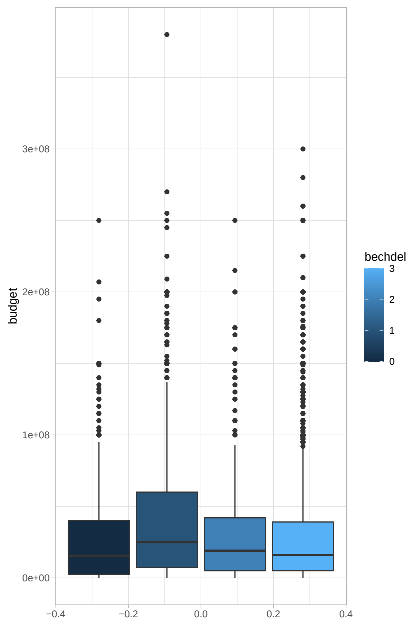
EDA

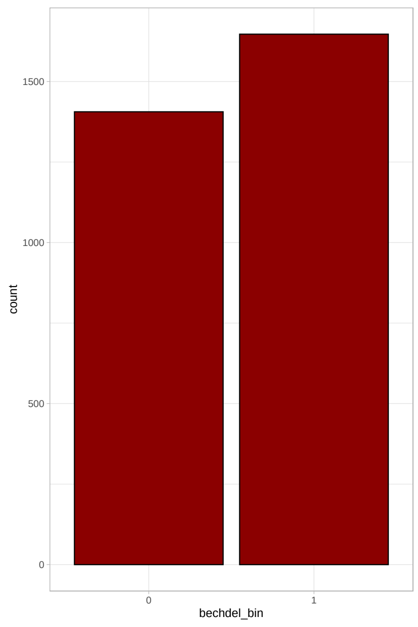












Additional Results

```
fit_gbm_bin$results %>%
  kable(digits = 3, caption = "Table: Stochastic Gradient Boosted Binary Classification") %>%
  kable_styling("striped", full_width = FALSE)
```

```
fit_knn_bin$results %>%
  kable(digits = 4, caption = "Table: Multinomial Multiclass Classification") %>%
  kable_styling("striped", full_width = FALSE)
```

Table 3: Table: Stochastic Gradient Boosted Binary Classification

	shrinkage	interaction.depth	n.minobsinnode	n.trees	ROC	Sens	Spec	ROCSD	SensSD	SpecSD
1	0.1	1	10	50	0.672	0.467	0.766	0.014	0.045	0.030
4	0.1	2	10	50	0.679	0.522	0.733	0.018	0.027	0.038
7	0.1	3	10	50	0.687	0.539	0.731	0.018	0.032	0.034
2	0.1	1	10	100	0.679	0.505	0.749	0.024	0.029	0.038
5	0.1	2	10	100	0.686	0.542	0.724	0.022	0.026	0.033
8	0.1	3	10	100	0.688	0.560	0.723	0.022	0.025	0.025
3	0.1	1	10	150	0.682	0.519	0.738	0.023	0.029	0.029
6	0.1	2	10	150	0.684	0.553	0.719	0.023	0.040	0.029
9	0.1	3	10	150	0.685	0.558	0.723	0.025	0.026	0.027

Table 4: Table: Multinomial Multiclass Classification

k	ROC	Sens	Spec	ROCSD	SensSD	SpecSD
5	0.5389	0.4474	0.6139	0.0202	0.0319	0.0235
7	0.5371	0.4147	0.6278	0.0297	0.0282	0.0344
9	0.5394	0.4125	0.6296	0.0245	0.0294	0.0195

```r
fit_logistic_bin$results %>%
  kable(digits = 3, caption = "Table: Random Forest Binary Classification") %>%
  kable_styling("striped", full_width = FALSE)
```

Table 5: Table: Random Forest Binary Classification

parameter	ROC	Sens	Spec	ROCSD	SensSD	SpecSD
none	0.685	0.541	0.724	0.024	0.03	0.026