Model building

Cross validation

Prof. Dr. Jan Kirenz

The following content is based on Mine Çetinkaya-Rundel's excellent book Data Science in a Box

Data and exploration


the office

Data

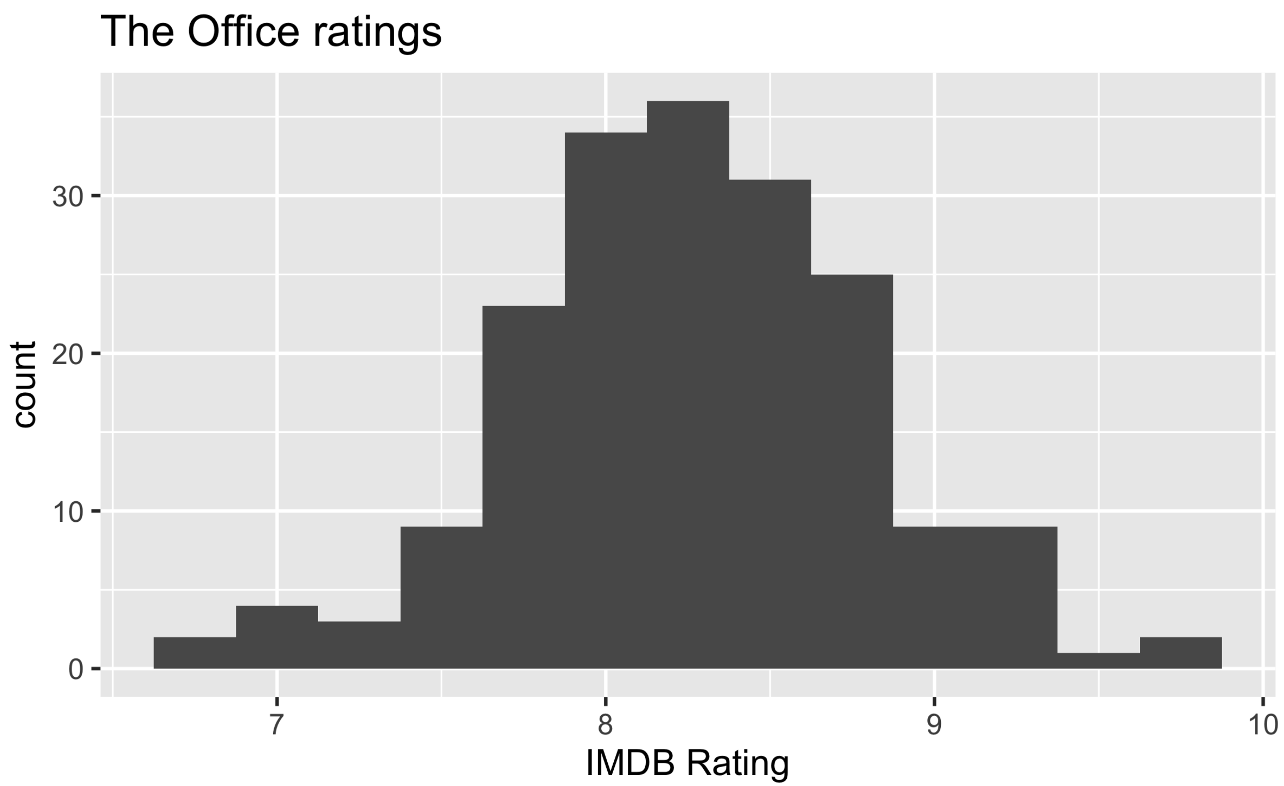
```
office_ratings <- read_csv("data/office_ratings.csv")
office_ratings
```

```
## # A tibble: 188 x 6
##   season episode title             imdb_rating total_votes air_date
##    <dbl>   <dbl> <chr>                   <dbl>       <dbl> <date>
## 1      1       1 Pilot                     7.6        3706 2005-03-24
## 2      1       2 Diversity Day             8.3        3566 2005-03-29
## 3      1       3 Health Care               7.9        2983 2005-04-05
## 4      1       4 The Alliance              8.1        2886 2005-04-12
## 5      1       5 Basketball                8.4        3179 2005-04-19
## 6      1       6 Hot Girl                  7.8        2852 2005-04-26
## # … with 182 more rows
```

Source: The data come from data.world, by way of TidyTuesday.

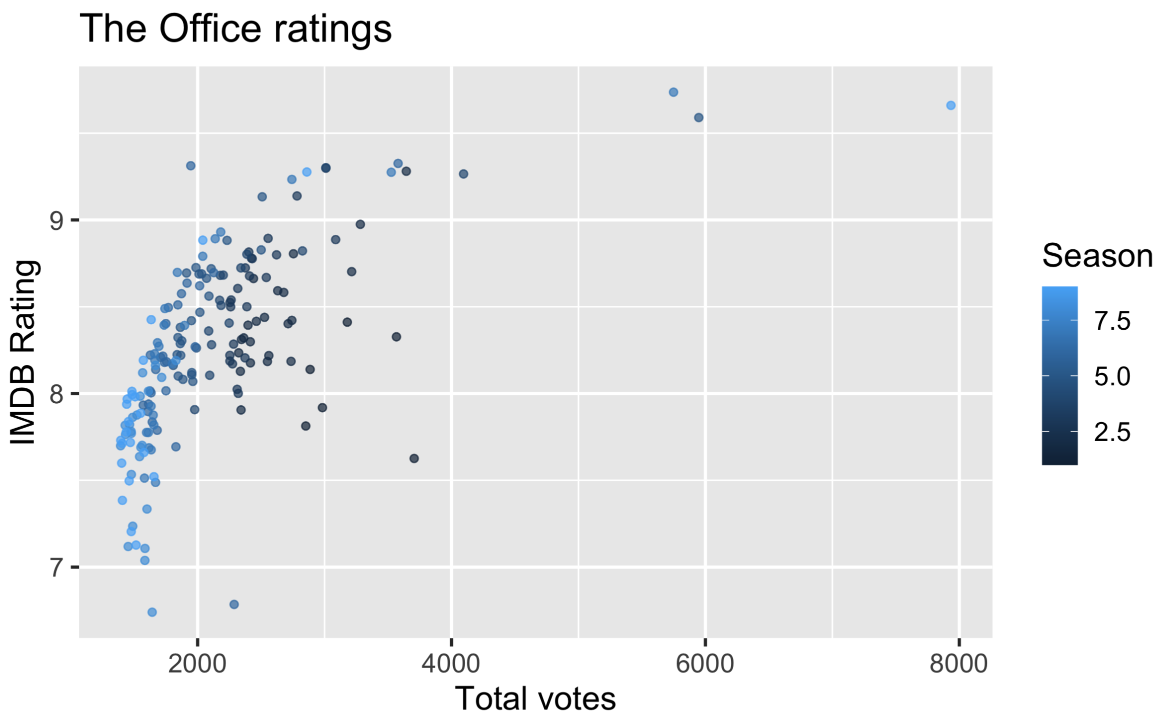
IMDB ratings

Code | Plot



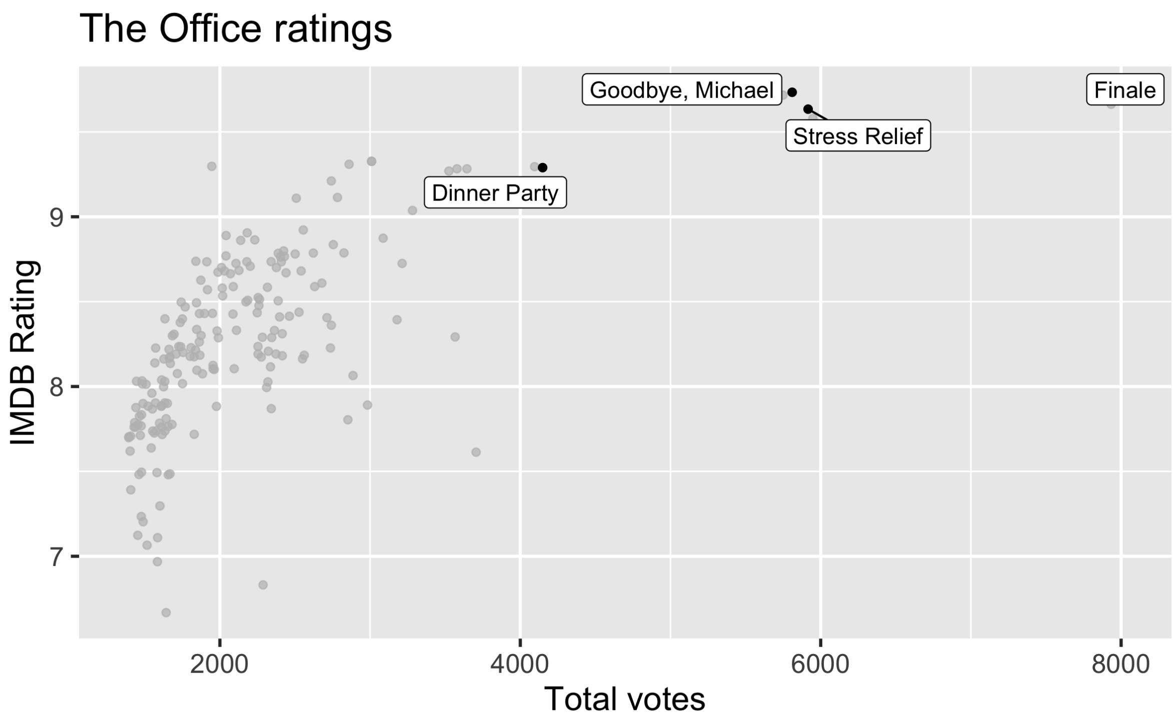
IMDB ratings vs. number of votes

Code | Plot



Outliers

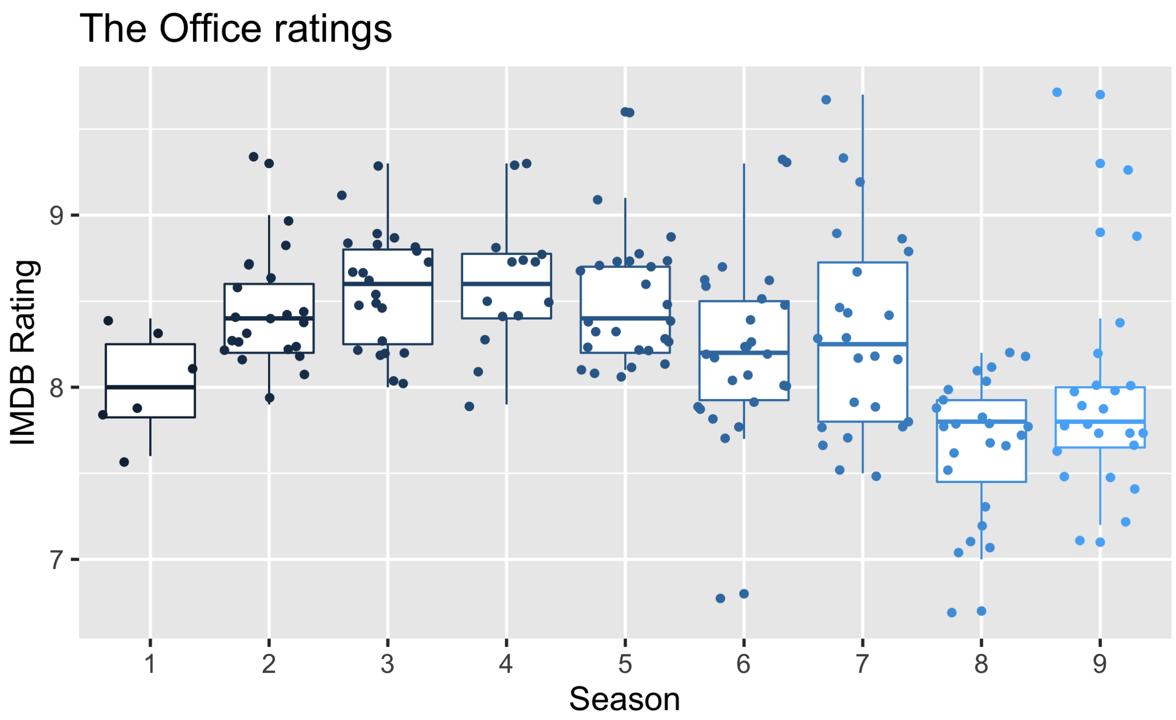
Code Plot



If you like the Dinner Party episode, I highly recommend this "oral history" of the episode published on Rolling Stone magazine.

IMDB ratings vs. seasons

Code Plot



Modeling

Train / test

- Create an initial split

```r
set.seed(1122)
office_split <- initial_split(office_ratings) # prop = 3/4 by default
```

- Save training data

```r
office_train <- training(office_split)
dim(office_train)
```

```
## [1] 141   6
```

- Save testing data

```r
office_test  <- testing(office_split)
dim(office_test)
```

```
## [1] 47  6
```

Specify model

```r
office_mod <- linear_reg() %>%
  set_engine("lm")

office_mod
```

```
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

Build recipe

Code Output

```
office_rec
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##         ID          1
##    outcome          1
##  predictor          4
##
## Operations:
##
## Date features from air_date
## Delete terms air_date
## Dummy variables from contains("month")
## Zero variance filter on all_predictors()
```

Build workflow

Code Output

```
office_wflow
```

```
## ══ Workflow ════════════════════════════════════════════════════════════════════
## Preprocessor: Recipe
## Model: linear_reg()
##
## ── Preprocessor ────────────────────────────────────────────────────────────────
## 4 Recipe Steps
##
## ● step_date()
## ● step_rm()
## ● step_dummy()
## ● step_zv()
##
## ── Model ───────────────────────────────────────────────────────────────────────
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

Fit model

Code Output

```
tidy(office_fit) %>%
  print(n = 12)
```

```
## # A tibble: 12 x 5
##    term                estimate std.error statistic  p.value
##    <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)         6.63     0.271       24.5    2.60e-50
##  2 season             -0.0297   0.0180      -1.65   1.02e- 1
##  3 episode             0.0453   0.00956      4.74   5.68e- 6
##  4 total_votes         0.000510 0.0000692    7.38   1.75e-11
##  5 air_date_month_Feb  0.00327  0.142        0.0229 9.82e- 1
##  6 air_date_month_Mär -0.0585   0.146       -0.402  6.89e- 1
##  7 air_date_month_Apr -0.150    0.141       -1.06   2.90e- 1
##  8 air_date_month_Mai  0.0231   0.178        0.129  8.97e- 1
##  9 air_date_month_Sep  0.646    0.180        3.58   4.80e- 4
## 10 air_date_month_Okt  0.572    0.154        3.71   3.11e- 4
## 11 air_date_month_Nov  0.349    0.140        2.49   1.41e- 2
## 12 air_date_month_Dez  0.516    0.158        3.27   1.39e- 3
```

Evaluate model

Make predictions for training data

```r
office_train_pred <- predict(office_fit, office_train) %>%
  bind_cols(office_train %>% select(imdb_rating, title))

office_train_pred
```

```
## # A tibble: 141 x 3
##   .pred imdb_rating title
##   <dbl>       <dbl> <chr>
## 1  8.48         7.6 Pilot
## 2  8.45         8.3 Diversity Day
## 3  8.10         8.1 The Alliance
## 4  8.30         8.4 Basketball
## 5  8.18         7.8 Hot Girl
## 6  8.90         8.7 The Dundies
## # … with 135 more rows
```

R-squared

Percentage of variability in the IMDB ratings explained by the model

```
rsq(office_train_pred, truth = imdb_rating, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.533
```

> Are models with high or low R^2 more preferable?

RMSE

An alternative model performance statistic: **root mean square error**

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

```
rmse(office_train_pred, truth = imdb_rating, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.350
```

> Are models with high or low RMSE are more preferable?

Interpreting RMSE

> Is this RMSE considered low or high?

```
rmse(office_train_pred, truth = imdb_rating, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.350
```

```
office_train %>%
  summarise(min = min(imdb_rating), max = max(imdb_rating))
```

```
## # A tibble: 1 x 2
##     min   max
##   <dbl> <dbl>
## 1   6.8   9.7
```

but, really, who cares about predictions on **training** data?

Make predictions for testing data

```r
office_test_pred <- predict(office_fit, office_test) %>%
  bind_cols(office_test %>% select(imdb_rating, title))

office_test_pred
```

```
## # A tibble: 47 x 3
##   .pred imdb_rating title
##   <dbl>       <dbl> <chr>
## 1  8.11         7.9 Health Care
## 2  8.68         8.2 Halloween
## 3  8.36         8.3 The Secret
## 4  8.56         8.1 Michael's Birthday
## 5  8.47         8   Grief Counseling
## 6  8.49         8.2 Initiation
## # … with 41 more rows
```

Evaluate performance on testing data

- RMSE of model fit to testing data

```
rmse(office_test_pred, truth = imdb_rating, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.482
```

- R^2 of model fit to testing data

```
rsq(office_test_pred, truth = imdb_rating, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.496
```

Training vs. testing

metric	train	test	comparison
RMSE	0.350	0.482	RMSE lower for training
R-squared	0.533	0.496	R-squared higher for training

Evaluating performance on training data

- The training set does not have the capacity to be a good arbiter of performance.
- It is not an independent piece of information; predicting the training set can only reflect what the model already knows.
- Suppose you give a class a test, then give them the answers, then provide the same test. The student scores on the second test do not accurately reflect what they know about the subject; these scores would probably be higher than their results on the first test.

Source: tidymodels.org

Cross validation

Cross validation

More specifically, **v-fold cross validation**:

- Shuffle your data v partitions
- Use 1 partition for validation, and the remaining v-1 partitions for training
- Repeat v times

You might also heard of this referred to as k-fold cross validation.

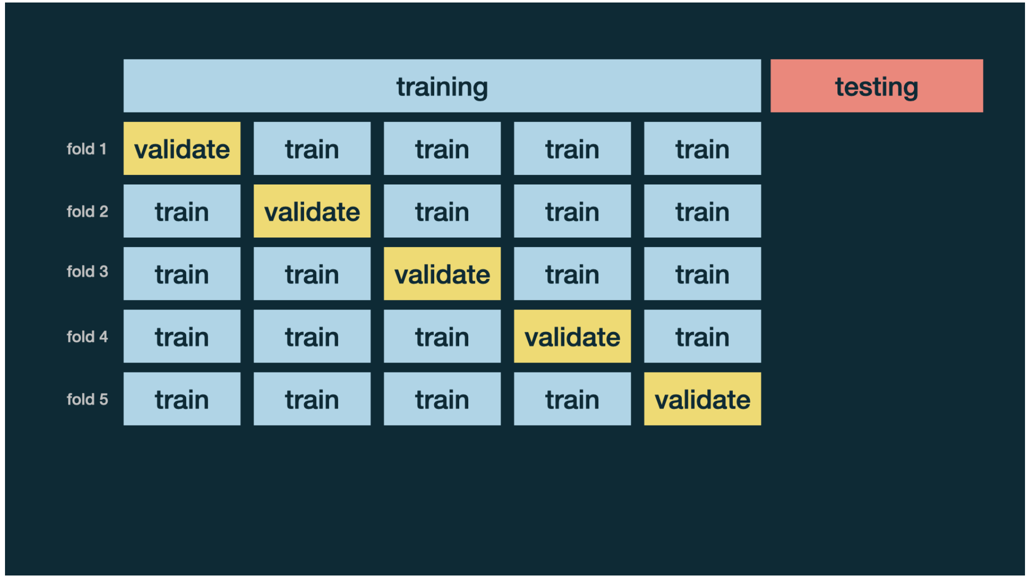
Cross validation

Split data into folds

```r
set.seed(345)

folds <- vfold_cv(office_train, v = 5)
folds
```

```
## #  5-fold cross-validation
## # A tibble: 5 x 2
##   splits            id
##   <list>            <chr>
## 1 <split [112/29]> Fold1
## 2 <split [113/28]> Fold2
## 3 <split [113/28]> Fold3
## 4 <split [113/28]> Fold4
## 5 <split [113/28]> Fold5
```



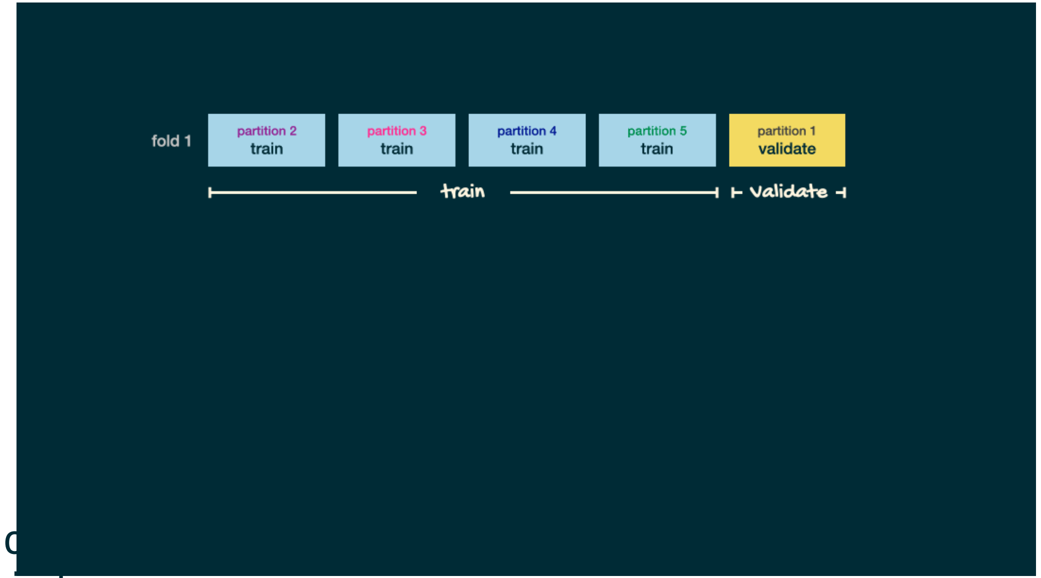
Fit resamples

```
set.seed(456)

office_fit_rs <- office_wflow %>%
  fit_resamples(folds)

office_fit_rs
```

```
## # Resampling results
## # 5-fold cross-validation
## # A tibble: 5 x 4
##   splits            id    .metrics         .no
##   <list>            <chr> <list>           <list>
## 1 <split [112/29]> Fold1 <tibble [2 × 4]> <tibble [0 × 1]>
## 2 <split [113/28]> Fold2 <tibble [2 × 4]> <tibble [0 × 1]>
## 3 <split [113/28]> Fold3 <tibble [2 × 4]> <tibble [0 × 1]>
## 4 <split [113/28]> Fold4 <tibble [2 × 4]> <tibble [0 × 1]>
## 5 <split [113/28]> Fold5 <tibble [2 × 4]> <tibble [0 × 1]>
```



Collect CV metrics

```
collect_metrics(office_fit_rs)
```

```
## # A tibble: 2 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   0.378     5  0.0260 Preprocessor1_Model1
## 2 rsq     standard   0.442     5  0.0816 Preprocessor1_Model1
```

Deeper look into CV metrics

Raw Tidy

Fold	RMSE	R-squared
Fold1	0.399	0.449
Fold2	0.306	0.749
Fold3	0.438	0.390
Fold4	0.419	0.280
Fold5	0.327	0.344

How does RMSE compare to y?

- Cross validation RMSE stats

```
## # A tibble: 1 x 6
##      min   max  mean   med     sd    IQR
##    <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>
## 1 0.306 0.438 0.378 0.399 0.0581 0.0918
```

- Training data IMDB score stats

```
## # A tibble: 1 x 6
##      min   max  mean   med    sd   IQR
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1   6.8   9.7  8.26   8.3 0.514 0.700
```

What's next?