What Makes a Review Valuable? - Predicting the Value of Yelp Reviews to Review Readers

November 22, 2015

Introduction

What makes a Yelp review valueable to review readers. The Yelp allows readers to rate reviews as useful, funny, or cool. This analysis uses these votes as a measure of review value: reviews with more reviewer votes should reflect the value of that review to readers.

Using the *Yelp Academice Dataset* I set out to predict the what makes a Yelp review more or less valuable than a different review. My goal was to create a model that did well and classifing reveiws but was easily understandable and could be used as guidance for review writers.

Findings Using a tree model I was able to correctly classify 65 percent of reviews in my validation data set correctly: A 16 percent improvement over a *no information* model.

The final model suggests that the content of reviews dominates review reader voting and that review writers should focus first on writting *useful* reviews and then *cool* reviews.

Methods

Data Collection

My analysis used the *Yelp Academic Dataset*. This dataset contains reviews, business, and user data from Yelp for four cities. It focused on reviews, reviewers, and reviewed businesses in the Yelp dataset. (Details about this dataset are available at the Yelp Dataset Challenge where the data set is available for download.)

I used the Yelp data to compute predictor and outcome variable (reader vote counts) and predictor data:

- **review data** - word counts, positive and negative sentiment word counts
- **reviewer data** - mean votes mean funny, cool, and useful votes per review, friend count, years as a Yelp Elite member, if they are a Yelp Elite 2015 member, and
- **business data** - whether the buisiness is open, average business star rating, and number of checkins the business has recevied.

```
## Loading required package: lattice
## Rattle: A free graphical interface for data mining with R.
## Version 4.0.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

Splitting the dataset

I partitioned the original data set into two separate data sets: a training set with about 80 percent of the observations, and validation data set containing about 20 percent of the observations. The validation data set was used to test the prediction accuracy of the final model.

The training data set was divided into *train dataset* and a *test dataset* with about 80 percent of the training cases being assigned to the *train dataset*.

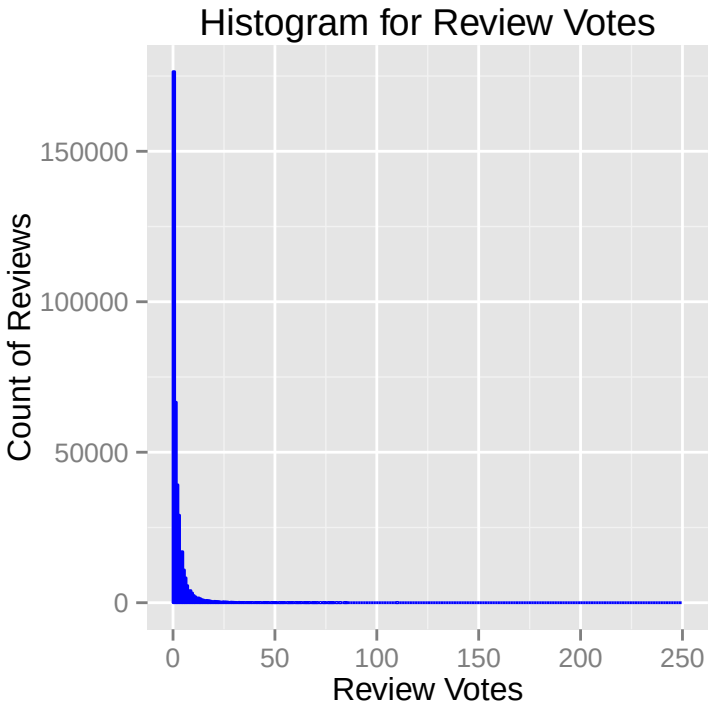
Exploratory Analysis

Variables of no practical value

First I removed variables of no practical value. These were variables like IDs and names of the businesses, review writers, and reviews.

Distribution of Review Votes

The distribution of review votes is highly skewed with over 40 percent of reviews getting zero votes. The distribution of votes also showed a long tail with reviews getting almost 250 votes.



I categorized review votes in to three categories: **low** - reviews receiving no votes (47% of the reviews), **moderate** - reviews receiving between 1 and 9 votes (49% of the reviews), and **high** - reviews receiving 10 or more votes (4% of the reviews).

```
##
##     high      low moderate
##     0.04     0.47     0.49
```

Predictor filtering

Elimiate near-zero varability covariates

I checked to see if the data had covariates with near-zero variability. The data did not have any, so no covariates were eliminated due to non-zero variability.

```
## character(0)
```

Elminate unneeded corvariates

Given a number of highly correlated predictors, I set out to eliminating unneeded covariates. I used the procedure suggested by Kuhn & Johnson (2013) p. 47 for reducing the effects of multicollinearity. Using this procedure, the *average votes a reviewer recieves* was removed. This left 16 predictor variables.

```
## [1] "average.votes"
```

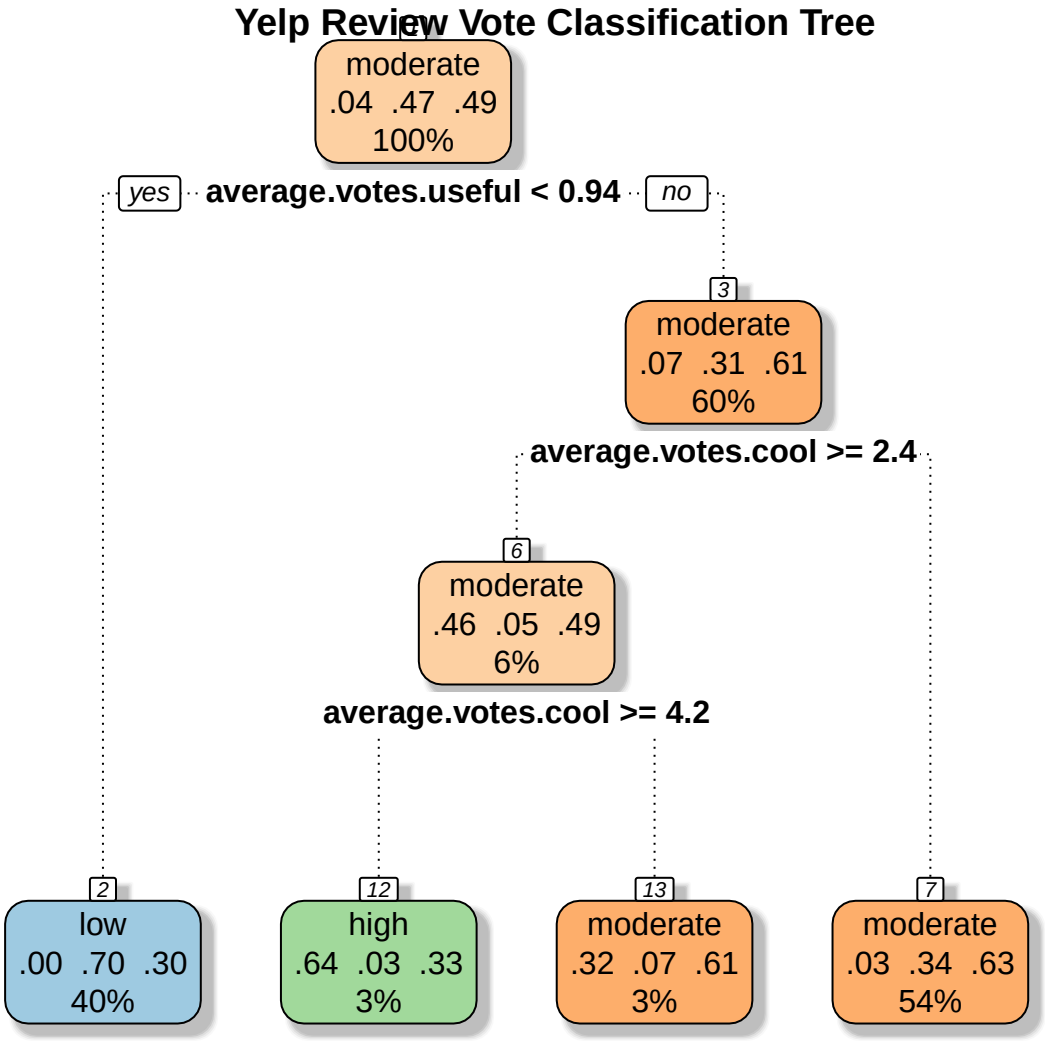
Correlation of remaining predictors

Train model

I used a tree model to predict reviewer vote categories (*low*, *moderate*, *high*). I selected a tree model because I wanted a model that could be easily understood by review writers and predict reivew votes that had a long-tailed distribution.

```
## Loading required package: rpart
```

```
## n= 878787
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 878787 449675 moderate (0.0431526638 0.4685469858 0.4883003504)
##    2) average.votes.useful< 0.945 355197 107901 low (0.0009037238 0.6962220965 0.3028741797) *
##    3) average.votes.useful>=0.945 523590 202058 moderate (0.0718138238 0.3140949980 0.6140911782)
##      6) average.votes.cool>=2.435 51668  26587 moderate (0.4634396532 0.0511341643 0.4854261826)
##       12) average.votes.cool>=4.165 23140   8348 high (0.6392394123 0.0337510804 0.3270095073) *
##       13) average.votes.cool< 4.165 28528  11014 moderate (0.3208426809 0.0652341559 0.6139231632) *
##      7) average.votes.cool< 2.435 471922 175471 moderate (0.0289369853 0.3428850530 0.6281779616) *
```




```
## pdf
##   2
```

Importance of predictors for simple model

```
##       Overall                names Proportion
## 1  65916.6280 average.votes.useful       0.26
## 2  64219.2005   average.votes.cool       0.26
## 3  54344.6334  average.votes.funny       0.22
## 4  36108.1185         friend.count       0.14
## 5  22819.8053          elite.count       0.09
## 6   6839.8187           word.count       0.03
## 7    536.2392           review.pos       0.00
## 8      0.0000         review.stars       0.00
## 9      0.0000           review.neg       0.00
## 10     0.0000            biz.stars       0.00
## 11     0.0000     biz.review.count       0.00
## 12     0.0000         biz.opentrue       0.00
```

```
## 13     0.0000 reviewer.review.count       0.00
## 14     0.0000        reviewer.stars       0.00
## 15     0.0000             elite2015       0.00
## 16     0.0000        yelping.months       0.00
```

Fit for the train data set The final tree model had about a 65.5 percent accuracy rate compared with a *no information* accuracy rate of 48.8 percent (a 16.7 percent improvement over *no information*). I accepted the model and applied it to the test data set.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   high    low moderate
##   high      14792    781     7567
##   low         321 247296   107580
##   moderate  22809 163676   313965
##
## Overall Statistics
##
##                Accuracy : 0.6555
##                  95% CI : (0.6545, 0.6565)
##     No Information Rate : 0.4883
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3517
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: high Class: low Class: moderate
## Sensitivity              0.39006     0.6006          0.7317
## Specificity              0.99007     0.7690          0.5853
## Pos Pred Value           0.63924     0.6962          0.6274
## Neg Pred Value           0.97297     0.6859          0.6956
## Prevalence               0.04315     0.4685          0.4883
## Detection Rate           0.01683     0.2814          0.3573
## Detection Prevalence     0.02633     0.4042          0.5695
## Balanced Accuracy        0.69007     0.6848          0.6585
```

Predictions for the test data set The tree model had a 65.5 percent accuracy rate for the test data. I accepted the model and applied it to the validation data set.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   high    low moderate
##   high       6237    335     3189
##   low         157 105411    46138
##   moderate   9857  70719   134577
##
## Overall Statistics
##
```

```
##                Accuracy : 0.6538
##                  95% CI : (0.6523, 0.6553)
##     No Information Rate : 0.4883
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3482
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: high Class: low Class: moderate
## Sensitivity              0.38379     0.5973          0.7318
## Specificity              0.99022     0.7687          0.5819
## Pos Pred Value           0.63897     0.6948          0.6255
## Neg Pred Value           0.97270     0.6841          0.6945
## Prevalence               0.04315     0.4685          0.4883
## Detection Rate           0.01656     0.2799          0.3573
## Detection Prevalence     0.02592     0.4028          0.5713
## Balanced Accuracy        0.68701     0.6830          0.6568
```

Predictions for the validation data set The tree model had a 65.6 percent did not show a drop in prediction accuracy. I accepted this as my model for submission and I next applied it to the 313,850 validation test cases. The tree model had an accuracy of 65.6 percent when predicting the validatation data. (*No information* had a 48.9 percent accuracy rate.)

Results

I created a tree model that was able to correctly classify about 65 percent of the reviewer votes categories in the validation data set. The table below presents the confusion matrix as proportions for the final tree model when applied to the validation data set.

high

low

moderate

high

0.02

0.00

0.01

low

0.00

0.28

0.12

moderate

0.03

0.19

0.36

The confusion matrix shows that most prediction confusions occurred when predicting *low* and *moderate* reviews with 87 percent of misclassifications being confusions between low and moderate reviews.

Discussion

Advice for Review writers

The final tree model suggests that the content of reviews dominates review reader voting and that review writers should focus first on writing *useful* reviews and then *cool* reviews. A follow-up question is "What makes a review cool?". I don't know. If review votes are cast after a review is read but before the reader has interacted with the business, *entertainment* may be a big factor. Clearly, what makes a review *cool* needs to be better understood.

Next Steps

- Find a better way to model data with a vast number of zeros and a wide range of values.
- Understand what distinguishes *cool reviews* from other reviews.

References

[1] Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling.* Springer: New York.

[2] The data set I used was originally downloaded from Yelp Dataset Challenge on November 15, 2015.

[3] Liaw, A., & Wiener, M., Classification and Regression by randomForest. R News: The Newsletter of the R Project, 18-22, http://cran.r-project.org/doc/Rnews/Rnews_2002-3.pdf, December, 2002.

[4] GEngelbeck (22 Nov. 2015). *Making Yelp Reviews Valuable to Readers?*. RPubs.

[5] GEngelbeck (22 Nov. 2015). Making Yelp Reviews Valuable to Readers?: R Project files