Predicting Closure of Businesses Based on Rating Behaviour

J.S. Ramos

November 20, 2015

1. Introduction

This study aims to classify and predict whether a business will permanently close or remain active based on the number of user ratings and the stars given by those within Yelp's yearly challenge. The data contains information on users, businesses, ratings and check-ins on these ratings.

Ratings are given at discrete moments in time with 1 to 5 stars. By building a predictive model that considers the change in number of ratings and stars across time we expect to arrive to a working classification model. Data on other entities like users and check-ins will not be considered for prediction exercise. Though this may hint at time-series modeling, powerful classification algorithms like bagging or random forest *are not suited for time series*, so instead of having ($\Delta_{ratings}$ and Δ_{stars}) across time, we will model them in yearly, discrete instants regardless of the chronological place of such years.

Finally, we will discuss the variable selection, accuracy, sensitivity and ROC curve of the prediction machine, and elaborate on alternative algorithms or variables.

2. Methods and Data

2.1. Obtaining and Cleaning Data

The original data is in JSON format, and consists of 5 files totaling about 1.6GB uncompressed. Since the only entities we're interested in are ratings (yelp_academic_dataset_review.json) and businesses (yelp_academic_dataset_business.json), we won't bother with other files. The cleaning process involves the following steps:

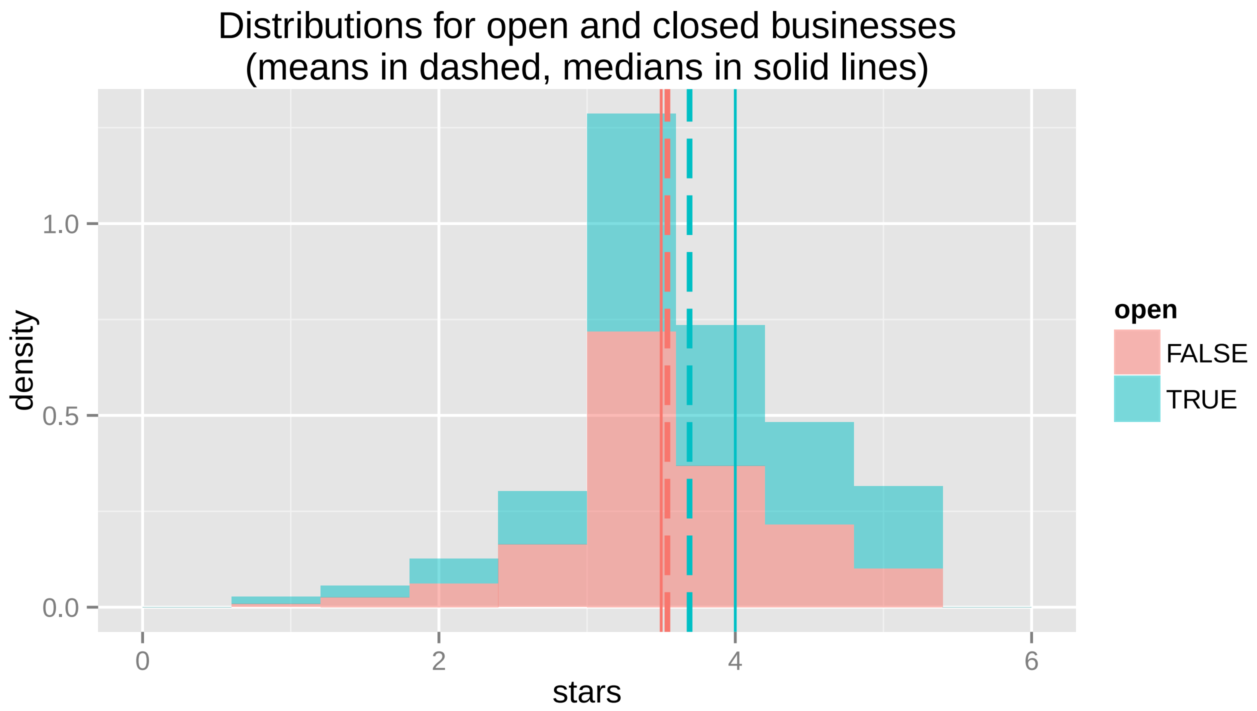
1. **Parse file contents and convert from JSON to data frame**: JSON arrays are enclosed in '[]' and each JSON object separated by ','. We use `flatten = T, simplifyVector = T, simplifyDataFrame = T` to maximize flattening of the data.
2. **Replace . (dot) in column names for _ (dash)**: dashes are more command-neutral than dots, and so preferred as column names.
3. **Drop columns that are not related to rating counts or stars granted**: we'll not perform any text-based sentiment analysis, nor address users nor check-ins. The scope of this study concerns itself with only review counts and stars granted.
4. **Convert date strings to POSIXct**: this is important in order to first have a time series and then convert it to discrete yearly observations as our method requires.

2.2. Exploring the Research Question with Hypothesis Testing

We are interested in predicting if a business will remain open or close based on the ratings it receives, so to explore if this question is worth answering with the data we have, we perform a *hypothesis test*, with the following hypotheses:

1. H_0: $\mu_{open} = \mu_{closed}$
2. H_a: $\mu_{open} <> \mu_{closed}$

To this end we first examine the difference in the distributions and means of each:



From the plot we can see that the mean stars received by each type of business is different, and that although the distributions are a little skewed to the left (given 5 max stars), they still follow an approximately normal curve. Also note the considerably lower number of ratings of any number of stars that closed businesses receive (i.e. a little less than half of those received by businesses that are still open for the mean number of stars).

This allows us to perform a Welch's 2-sample T-test to assess which hypothesis is valid.

```r
# Welch T test for stars for open and closed businesses
hyptest <- t.test(open, closed, var.equal = F)
hyptest
```

```
##
##  Welch Two Sample t-test
##
## data:  open and closed
## t = 14.843, df = 10256, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.1297573 0.1692451
## sample estimates:
## mean of x mean of y
##  3.691531  3.542030
```

With a p-value of **0** and given that the 95% CI of **(0.13, 0.17)** is not centered around 0, we *fail to reject* H_0 and conclude that the question is worth studying further.

2.3. Converting a Time-Series to Wide Format

As explained above, $\Delta_{ratings}$ and Δ_{stars} are time series, but powerful statistical learning algorithms for classification are not suited for this type of data. Also, there are several businesses with only 3 ratings, meaning that we would only have 2 data points for these Δs. Moreover, ratings are not given at the same time for every business observed (i.e. a biz may open in '04 and receive its 1st review in '07, while another may open in '08 and receive its 1st review the same year). Due to these limitations, we must convert the review data set to a *wide format* and then append it to the business data set, a process that took the following steps:

1. **Partition the review count and stars by year**: we discretely partition the time series into yearly slices, so we need to sum the number of reviews and average the stars per year for each business.

```
##              business_id review_count_year review_stars_year
## 1 vcNAWiLM4dR7D2nwwJ7nCA                 1                 5
## 2 vcNAWiLM4dR7D2nwwJ7nCA                 1                 2
## 3 vcNAWiLM4dR7D2nwwJ7nCA                 3                 4
```

2. **Calculate $\Delta_{ratings}$ and Δ_{stars}** for each 2-year period with the `lag()` function to arrive to the following:

```
##              business_id review_count_change_year review_stars_change_year
## 1 vcNAWiLM4dR7D2nwwJ7nCA                       NA                       NA
## 2 vcNAWiLM4dR7D2nwwJ7nCA                        0                       -3
## 3 vcNAWiLM4dR7D2nwwJ7nCA                        2                        2
```

3. **Make the data wide**: since we cannot have repeated `business_id` because we wish to have a single classification (open = TRUE | FALSE) for each, we need to go from long format to wide format. See here for an explanation between these forms and their respective use cases.
4. **Join this new wide dataset with the business dataset**: by doing this we consolidate our data and remove 399 business observations that have no ratings data. Also, the key column `business_id` will be dropped since, being an id, it has 100% variance and of no interest to the prediction exercise. Finally, we'll only leave the variables that represent change (Δ), and drop those representing discrete states.

This process results in a dataset that contains **12 variables** indicating **12 years** of $\Delta_{ratings}$ and Δ_{stars}. Note that even though our process has created 12 vars, not all businesses have data for all the 12-year span, and this high number of 0s will definitely have a negative impact on the prediction machine.

2.4. Random Forest

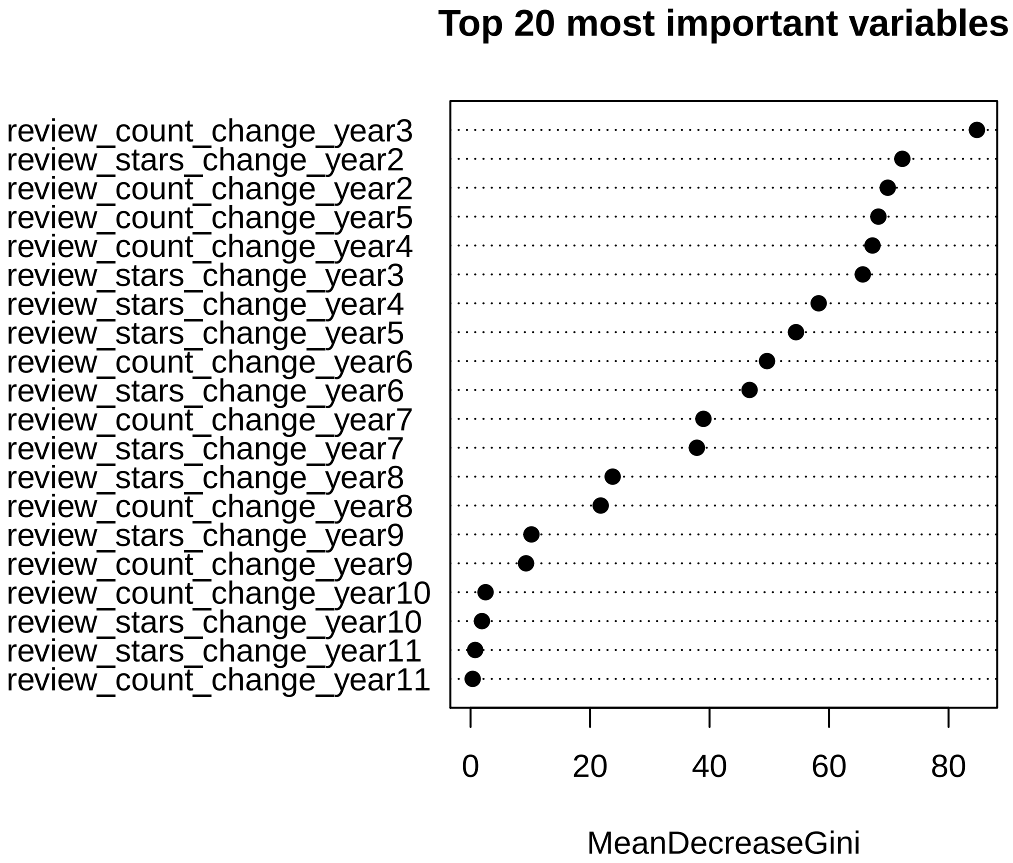
We choose a Random Forest model due to it having built-in feature selection. According to the `caret` package documentation, its feature selection algorithm is coupled with the parameter estimation algorithm, making it faster than if the features were searched for externally.

Also, even though some authors state that RFs do not require cross-validation, we will nonetheless train our RF with a 5-fold, single-pass cross-validation in order to reduce any potential bias. Our data sets will be 70% training and 30% test.

```
## Random Forest
##
## 42550 samples
##    24 predictor
##     2 classes: 'FALSE', 'TRUE'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 34040, 34040, 34039, 34041, 34040
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa       Accuracy SD   Kappa SD
##    2    0.8778378  0.00000000  5.832861e-05  0.000000000
##   13    0.8724324  0.04336314  1.636429e-03  0.009751225
##   24    0.8682491  0.06292241  2.233660e-03  0.007992067
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

2.4.2. Variable Selection & In-sample Error

The algorithm's built-in feature selection determined that `review_count_change_year3` is the variable that most helps to reduce impurity, followed by `review_stars_change_year2`. The possible interpretations we can arrive to from such plot is that 1) it would be possible to build a simpler model with review count and stars from years 2 to 6 without terribly affecting sensitivity or accuracy, and 2) that having a considerable change in review count and stars during the first years of operations may contribute in the long run to the continuity or shutdown of a business.



However, this interpretation falls short when we take a look at the model's accuracy and in-sample error.

```
##       FALSE  TRUE class.error
## FALSE     0  5198           1
## TRUE      0 37352           0
```

With an accuracy of 50% and an in-sample error of 50%, we realize that this high rate will not get better when applying it to the test set, since it is expected the out-of-sample error to be greater due to unaccounted-for bias in the test set, even if preemtively addressed with cross validation. Even though we have had a glimpse of how inadequate the model will be, we will do the test set for the sake of completeness.

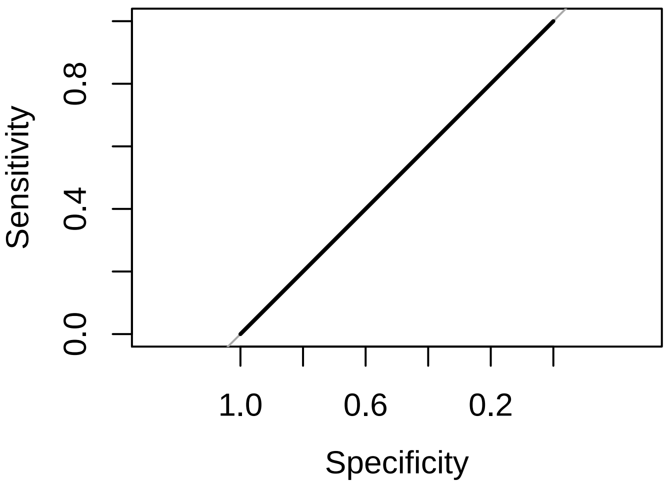
3. Results

We'll now apply the model to the test set and comment on the accuracy, specificity and out-of-sample error.

```
##           Reference
## Prediction FALSE  TRUE
##      FALSE     0     0
##      TRUE   2227 16008
```

From the confusion matrix we can surmise that the statistical machine does a fair job predicting that a business will remain open (with an accuracy of `87.79`%), but a very poor job at detecting when a firm will go bust (as evidenced by the 0% Sensitivity of the model).

To confirm this conclusion, we plot the ROC curve and see that the algorithm's positive predictive power is 0, so even if it can predict businesses that will remain open, the fact that it cannot predict the opposite makes the prediction machine unusable.



Call: plot.roc.default(x = rfModel*pred*obs[selectedIndices], predictor = rfModel*pred*mtry[selectedIndices])

Data: rfModel*pred*mtry[selectedIndices] in 5198 controls (rfModel*pred*obs[selectedIndices] FALSE) < 37352 cases (rfModel*pred*obs[selectedIndices] TRUE). Area under the curve: 0.5

In sum, we realize that the model is not the best-suited for the problem, and this could be attributed to any, or all, of the following causes:

1. As stated earlier, Random Forest Algorithms are not suited for time-based data, and our transformation of the data in 'lags', though correct, may have diluted the features that identify a business that had to shut down to the point that the algorithm cannot discern between one set and the other, as evidenced by the high number of 0s in several of the columns.
2. Given that the algorithm identified the variables and values as being the same for both types of businesses, it could be that the real classifying variables lie in another dataset, such as tips to the ratings (which would increase their weight), or check-ins to the businesses (since consuming from them is a more relevant indicator of activity than a subjective rating).

4. Conclusion

It is clear that the main lessons from this exercise are that Random Forest may not be the best option for binary classification, and that the necessary transformations to make it a viable option will, depending on the dataset, result in dilution of information to such degree that the data becomes impossible to classify. Another lesson is that further variable exploration may be required to arrive to a working machine, such as using check-ins instead of ratings, since the former is a clearer identifier of activity than the latter. But perhaps the most relevant lessons are those that form the core of the true work of the data scientist: 1) that the data cleaning process will take you 80%

of your time -as evidenced by the bulk of this work being data processing to go from time series to discrete lags, 2) that machine learning algorithms are often overkill and will never make up for initial exploration with GLMs, and 3) all models are wrong, but some are useful, and in this case, even though this model was wrong, it was still useful to steer further research into other, perhaps more relevant, variables.