How many “useful” votes will a Yelp Review Receive

Gustavo Sandoval, July 22, 2013

1 ABSTRACT

At Yelp they track 3 community-powered metrics of review quality: **Useful, Funny, Cool**. Over time, a good review will accumulate lots of votes in these categories from the community. However, another extremely important quality feature is the **freshness of a review**.

What if we didn't have to wait for the community to vote on the best reviews to know which ones are high quality? This project is concerned with **estimating** the number of useful votes a review will receive based on different attributes of the review.

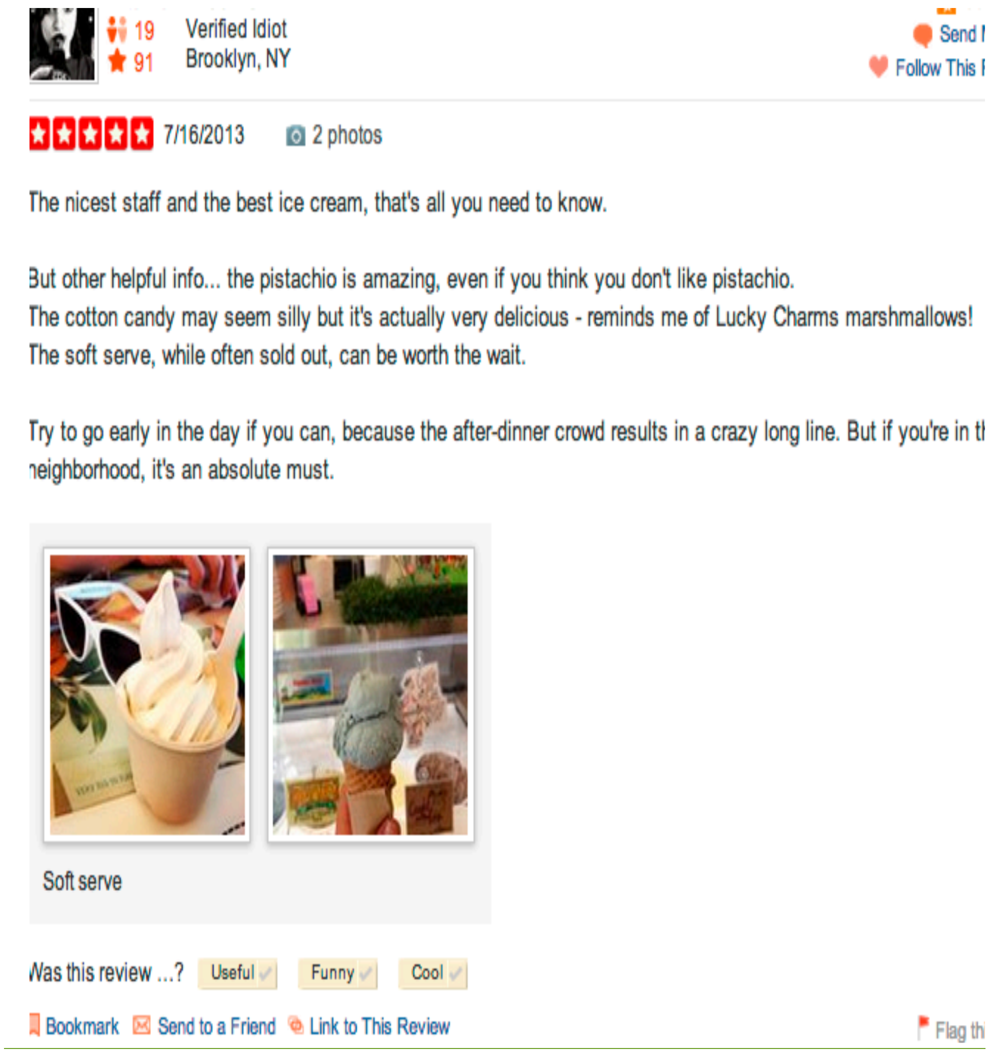
2 FORMULATION OF PROBLEM

This project is concerned with a Kaggle competition sponsored by Yelp. This competition ran from March 27th to June 30th. The competition problem statement was:

> At Yelp they track 3 community-powered metrics of review quality: **Useful, Funny, Cool**. Over time, a good review will accumulate lots of votes in these categories from the community. However, another extremely important quality feature is the **freshness of a review**. What if we didn't have to wait for the community to vote on the best reviews to know which ones are high quality? This project is concerned with **estimating** the number of Useful votes a review will receive.

The dataset given was Training data in the form of 229K reviews of 19K business and check-ins from 43K users. A link to the yelp competition is here: Yelp's Recruiting Competition.

In case the reader is not familiar with yelp (www.yelp.com): Yelp provides user reviews and recommendations for Restaurants, Shopping, Nightlife, Entertainment and others. This specific problem is concerned with the user reviews. For example here are two reviews from yelp for the same place: “Ample Hills Creamery”, an Ice Cream Shop in my neighborhood of Prospect Heights.



19
91
Verified Idiot
Brooklyn, NY

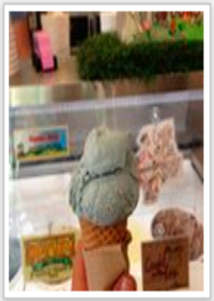
Send I
Follow This I

7/16/2013 2 photos

The nicest staff and the best ice cream, that's all you need to know.

But other helpful info... the pistachio is amazing, even if you think you don't like pistachio.
The cotton candy may seem silly but it's actually very delicious - reminds me of Lucky Charms marshmallows!
The soft serve, while often sold out, can be worth the wait.

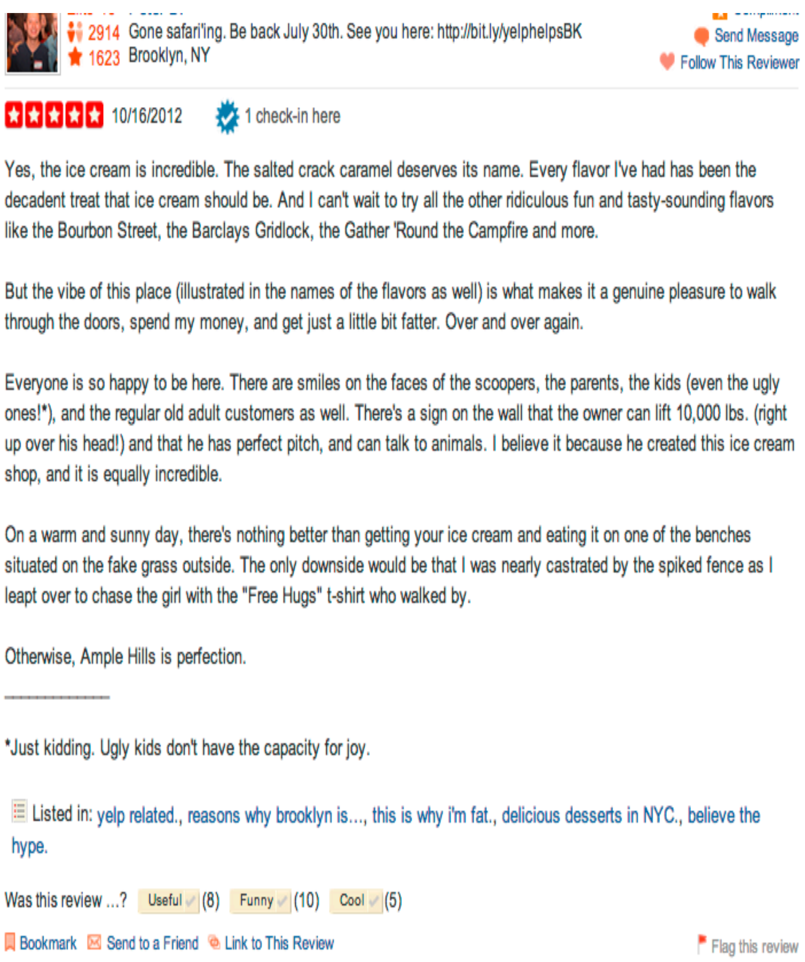
Try to go early in the day if you can, because the after-dinner crowd results in a crazy long line. But if you're in tl
neighborhood, it's an absolute must.



Soft serve

Was this review ...? Useful Funny Cool

Bookmark Send to a Friend Link to This Review Flag th



2914 Gone safari'ing. Be back July 30th. See you here: http://bit.ly/yelphelpsBK

1623 Brooklyn, NY

Send Message

Follow This Reviewer

10/16/2012 1 check-in here

Yes, the ice cream is incredible. The salted crack caramel deserves its name. Every flavor I've had has been the decadent treat that ice cream should be. And I can't wait to try all the other ridiculous fun and tasty-sounding flavors like the Bourbon Street, the Barclays Gridlock, the Gather 'Round the Campfire and more.

But the vibe of this place (illustrated in the names of the flavors as well) is what makes it a genuine pleasure to walk through the doors, spend my money, and get just a little bit fatter. Over and over again.

Everyone is so happy to be here. There are smiles on the faces of the scoopers, the parents, the kids (even the ugly ones!*), and the regular old adult customers as well. There's a sign on the wall that the owner can lift 10,000 lbs. (right up over his head!) and that he has perfect pitch, and can talk to animals. I believe it because he created this ice cream shop, and it is equally incredible.

On a warm and sunny day, there's nothing better than getting your ice cream and eating it on one of the benches situated on the fake grass outside. The only downside would be that I was nearly castrated by the spiked fence as I leapt over to chase the girl with the "Free Hugs" t-shirt who walked by.

Otherwise, Ample Hills is perfection.

---

*Just kidding. Ugly kids don't have the capacity for joy.

Listed in: yelp related., reasons why brooklyn is..., this is why i'm fat., delicious desserts in NYC., believe the hype.

Note how the first review doesn't have any useful votes, whereas the second one has 8 useful votes and one checkin.

3 Description of Data Set

The training data set consists of 4 different collections of data:

- Business (11,537 rows)
- User (43,873 rows)
- Checkin (8,282 rows)
- Review (229,907 rows)

We were given JSON files for each of the 4 data sets. The data sets are huge with the review data toping at 216MB of data. The others were far behind, but still big: User 7 MB, business 4 MB, and checkin at 3.5.

For each of the collections, here is the schema.

3.1 BUSINESS

```
'business_id': (encrypted business id),
'name': (business name),
'neighborhoods': [(hood names)],
'full_address': (localized address),
'city': (city),
'state': (state),
'latitude': latitude,
'longitude': longitude,
'stars': (star rating, rounded to half-stars),
'review_count': review count,
'categories': [(localized category names)]
'open': True / False (corresponds to permanently closed, not business
hours),
```

3.2 REVIEW

```
'business_id': (encrypted business id),
'user_id': (encrypted user id),
'stars': (star rating),
'text': (review text),
'date': (date, formatted like '2012-03-14'),
'votes': {'useful': (count), 'funny': (count), 'cool': (count)}
```

3.3 USER

```
'user_id': (encrypted user id),
'name': (first name),
'review_count': (review count),
'average_stars': (floating point average, like 4.31),
'votes': {'useful': (count),  'funny': (count), 'cool': (count)}
```

3.4 CHECKIN

```
'business_id': (encrypted business id),
'checkin_info': {

'0-0': (number of checkins from 00:00 to 01:00 on all Sundays),
'1-0': (number of checkins from 01:00 to 02:00 on all Sundays),
...
'14-4': (number of checkins from 14:00 to 15:00 on all Thursdays),
...
'23-6': (number of checkins from 23:00 to 00:00 on all Saturdays)
```

4 DESCRIPTIVE FEATURES:

4.1 DATA PREPARATION:

For preparing the data I tried a few approaches:

1. I first tried importing the data using the json package in python. This didn't work very well, because the parser crashed with the file with some errors. I tried a couple of other things but since they also failed, I moved on to another approach.
2. I tried importing the data into MongoDB using mongoimport. This worked very well right from the start. This is the way I wrote all of my code. The drawback is that I have a dependency on mongo and also the MongoClient python library.
3. Another approach that I considered in order to improve dependencies would be converting the JSON to CSV and then using Pandas to read the CSV file. This would avoid the Mongo dependency. I also implemented this, but in the end in the interest of time I went with the Mongo approach.

4.2 DESCRIPTIVE FEATURES:

From all the above features, I ended, using just a subset. The following is the subset used from each table:

- **Review:**
  - Date
  - Stars
  - Text
- **User:**
  - Average Number of stars given
  - Cool votes
  - Funny Votes
  - Useful Votes
- **Business:**
  - Count of reviews
  - Stars

In addition to the intrinsic data features, I ended up computing a few features from the data. The features were:

- **Computed**

- **Review star delta:** This is the delta from the user's star review and the business's star review.

- **Date Delta from 1/1/13:** This is the freshness of the review. It was computed form 1/1/13 in order to normalize between runs.

- **Text Length:** The character count of the review.
- **Text Word Count:** the number of words in the review.
- **Useful Ratio:** ratio of useful reviews to the number of total reviews per user.

4.3 DATA DISTRIBUTION:

After formulating a hypothesis on which would be the descriptive features. I created graphs plots for them. The following figures show the distributions of some of the most interesting ones along with the some statistics:

Figure 1: Count of Useful Votes on Reviews

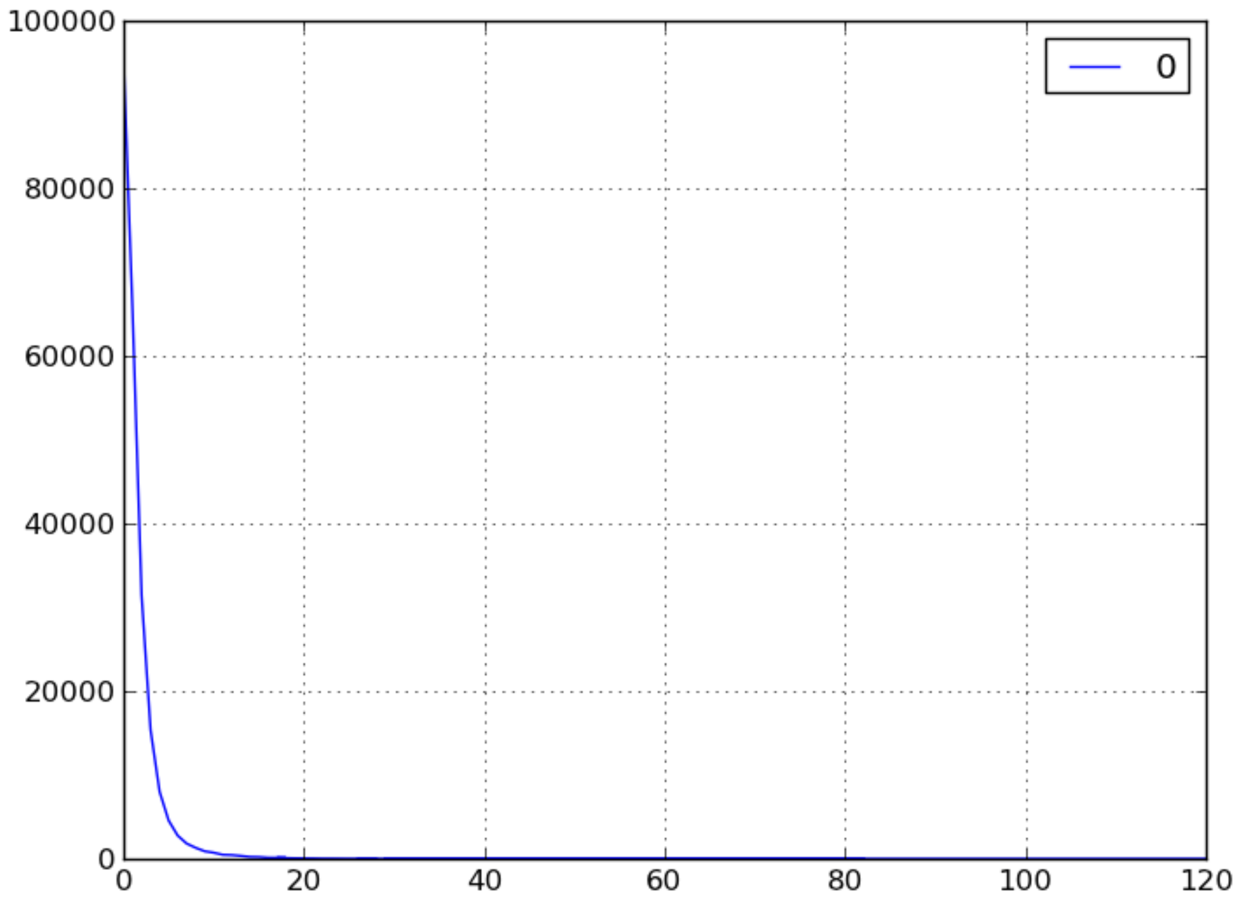
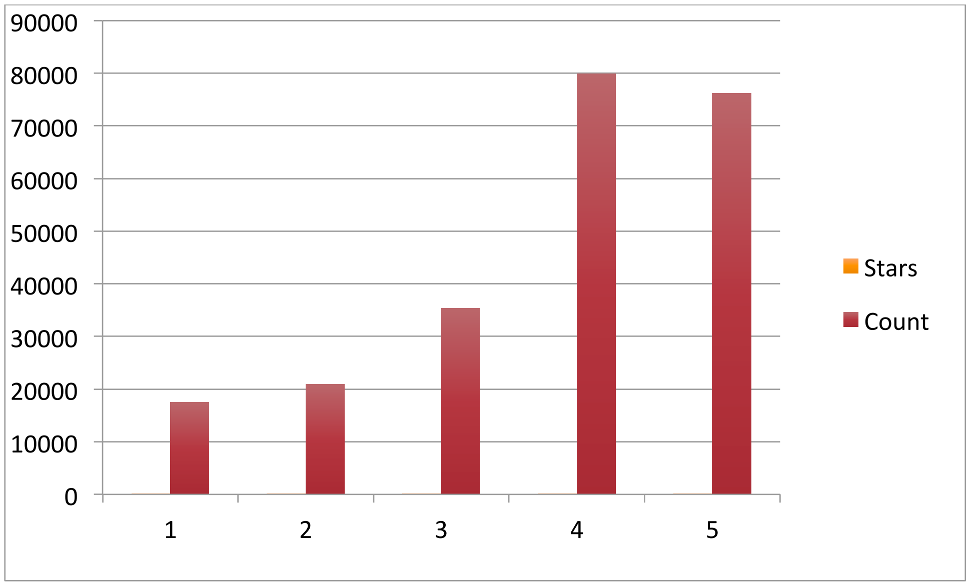


Figure 2: Star Distribution of Reviews



5 STATISTICAL METHODS

This is a supervised learning problem because we are trying to predict or forecast a target value

It's also a continuous value, so it’s a regression problem. Because of this we have two options for algorithms: multivariate regression or Random Forests. When I tried Multivariate regression, I didn’t get too far. So I tried Random Forests. I used Python’s Scikit learn implementation of Random Forests.

Here’s a summary of the methodology:

Data Preparation:

1. Import the JSON data into Mongo.
2. Import the Data from Mongo into Pandas DataFrames for the following Tables: Review, User and Business. The checkin data didn’t seem very helpful.

Data Analysis:

3. Print and graph the distributions of the data for each table
4. Perform a join on the Review and User Tables with ‘user_id’ as the key.
5. Join the result of the previous join with the Business Table with the ‘business_id’ as the key
6. Drop all the rows that had ‘Not a number’ or Nan
7. Calculate the Computed columns described in section 4.2: **Review star delta, Date Delta, Text Length, Text Word Count, UsefulRatio**
8. Divide the datasets into Training and Testing

Train:

9. Run the RandomForestAlgorithm with the Training data. The RandomForesRegressor was run with 150 trees as this was a good balance between computation time and results. More trees would only increase time while not increasing accuracy.

Test:

10. Run the algorithm with the Test data
11. Evaluate the performance of the algorithm using the root mean square log error (RMSLE)

6 CONCLUSIONS

After running the Random Forest algorithm, I was able to see how much each of the features weighed into the prediction. There are two interesting things to note.

1. if we don't include the Useful Ratio, then the best predictors are mostly how many funny votes the review got, followed by how many cool votes it got and finally how many reviews the user has written. This analysis got an RMSLE of 0.4797. The summary of the data looks as follows:

features	importance
review_dateDelta	0.90%
review_stars	0.23%
review_text_len	2.13%
review_word_count	0.17%
user_avgStars	0.17%
user_cool	7.17%
user_funny	84.73%
user_reviews	3.76%
user_useful	0.71%
biz_review_count	0.02%
biz_stars	0.00%
biz_review_star_delta	0.01%

2. However, if we include the useful ratio column that was computed between the number of useful reviews and the number of total reviews written, then this column completely eclipses everything else at 91%. This analysis got an RMSLE of **0.4791**. Here's the full data:

features	importance
review_dateDelta	4%
review_stars	0%

review_text_len	2%
review_word_count	0%
user_reviews	3%
user_useful	91%
biz_review_count	0%
biz_stars	0%
biz_review_star_delta	0%

In conclusion the best predictor of how many useful votes a review will get are attributes of the reviewer. The most important is the ratio of useful votes per review. In summary the quality of the reviews written by the user are the best predictor of how many useful votes a review will get. Another thing to note is that the RMSLE for the winner on the Kaggle competition was 0.4404 and mine was 0.4791 which put me in 52nd place out of 352 competitors.

7 Business Applications

The findings that the attributes of the reviewer are the best predictor of how many 'useful votes' a review will get are many. For example:

1. The yelp website can "showcase" users that have written many reviews or have gotten many useful votes since their reviews are very likely to get useful votes.
2. Yelp can hire writers to do reviews and make sure that they get showcased appropriately.
3. User's can look for reviews done by "powerusers" of the site. Ie. Folks that have written many reviews.

8 References and resources

8.1 APIs and software packages

All programming work was done in Python. The software packages that I used were:

- **NumPy (http://www.numpy.org/)** - Popular package for doing general mathematical work.
- **Pandas (http://pandas.pydata.org/)** - Offers data structures and functions specific to statistical work. The dataframe datastructure is very handy since it provides SQL type joins.
- **scikit-learn** (http://scikit-learn.org/stable/) - Offers a variety of algorithms and useful functions for machine learning, prediction, error checking and validation, and many others.
- **Matplotlib (http://matplotlib.org/)** a 2d python graphing library was used for some of the graphics on the code and on this paper.

- **Pymongo** (http:// http://api.mongodb.org/python/current/) - offers the tools for working with MongoDB from Python.

All the data was parsed into a local MongoDB. In addition to Mongo, I also used a GUI for Mongo called MongoHub (https://code.google.com/p/mongohub/), to see the data visually.

Microsoft Excel was also used to prepare some of the graphics in this paper.

9 BIBLIOGRAPHY

[1] "Machine Learning in Action". Harrington, Peter. April 2012. Manning Publications.