**TA's Solution**

This is the solution which we used and is available on leaderboard under the team name "TravisBickle".

Features used:

* outOf
* Rating
* Price (substituted by category wise average where NaN)
* Log of length of review
* [ARI](https://en.wikipedia.org/wiki/Automated_readability_index)

All the features were transformed using [PolynomialFeatures](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html) degree 2 and an ablation experiment was performed to come up with the best features.  As you can see, I did not use any major textual features.

Breaking down data (long tail problem):

* outOf values = 0 [Predict zero]
* outOf values between 7 and 15 (both inclusive)
* outOf values 16 and above

outOf values between one and 6 were ignored by me because they had too much noise. These values were predicted upon using the second model above. Note that this may not be the best way to deal with this data, but was something that worked well on my validation sets. An improvement can be modeling outOf = 1 as a classification problem.

Gradient Boosting Regressors with MAE criterion were used everywhere. Since this is the loss we wish to minimize, this criterion produces very good results.

**What can be done to improve**

* Model outOf = 1 as a classification problem [Ensembling of multiple classifiers may be used]
* Add features such as those described below
* Create multiple models for the outOf range ignored
* Identify the bottleneck of this model (the outOf bucket with max MAE) and work on this bucket alone first. You may modify the features and/or break it down further, based on the outOf value
* Use test set to your advantage - restrict training data to the range of values available in test data. This removes outliers and reduces data size, which is sometimes helpful in creating a more accurate model
* An ensemble of various regressor models combined via either majority vote, average or weighted average.
* For some datasets (not this dataset), it may be the case that price for an item is NaN in one row and available at another. Similarly, you may have the price for some item in the test set and not in the training set. Such substitutions and analysis may help here.
* Use [XGBoost](https://xgboost.readthedocs.io/en/latest/)instead of GBM (for both classification and regression). It is one of the best libraries that is around today.
* Neural networks

**What Works Well**

There are other things that we know work well but we did not use for this model. These include

* Making a good set of predictions (say, using the above method) and using the (predictions + training data) to create the average user rate feature. This is likely to perform better.
* Sentiment Analysis. Note that not all sentiments help. Some like polarity/sensitivity might help.
* Using other ensemble methods or using GridSearchCV with a custom MAE evaluator.
* Cross-validation - lots and lots of it. Data like this can be skewed in a single validation set.
* Square of number of reviews user has written - please use train+test set to determine this.

**What doesn't work [in most cases]**

* Very high number of features
* Directly using user's average helpfulness as a feature
* Having highly correlated features all at once - For example, number of sentences, length of review, number of words etc. are all strongly correlated. Having just one of them should be enough.

**General Feedback**

* How I like to approach regression problems like this is - use Least Squares Regression to come up with a list of good features. This helps because this model doesn't have any parameters to tune and the bad scores of my model do not depend upon me not choosing good parameters
* We have said this repeatedly - A model is only as good as its features. If your features don't convey you enough information about the target variable, you can never guess it correctly. Changing models and trying out different classifiers/regressors will help little when the features are not good enough.
* When predicting a ratio in regression models, always put explicit checks for < 0 and > 1. This is because an outlier on a single highly weighted feature can run havoc and end up giving you an abnormally high MAE/MSE.
* If you wish to try out something else on this problem, you can still submit more entries to the competition (non-graded) by going to the same Kaggle link
* We request everyone to go to "My Submissions" tab in Kaggle and see how each one of their entries fared on public and private leaderboards.
* In almost all data science competitions, an ensemble method comes out on top. It may take longer to train/predict, but since competitions like this care only about the final submission file, ensembling is the perfect way to go
* Rounding decreases MAE in problems where target variable is an integer (provided your model is good). We noticed that many people did not round their results. This point was mentioned in hint #4
* When submitting a solution to Kaggle, it is advisable to write a summary of your model plus the validation score it achieved for you in the description of the solution. This is extremely helpful while selecting the final submissions.
* Always try to use test set to your advantage, as mentioned above in a few points

We hope that you enjoyed the competition and will participate in public Kaggle challenges during your free time to sharpen your skills. An added source of motivation - one of the recent [Kaggle challenges by Zillow](https://www.kaggle.com/zillow/competitions) has a prize money of over a million US dollars!