Kevin Chang (A10031680)
William Chen (A10110902)
Kevin Yang (A10798495)
CSE 190, Fall 2015
McAuley
Assignment 2

Rating Prediction on the Amazon Movie Dataset using Various Regression Techniques

Introduction (1)

Users on e-commerce sites like Amazon write reviews and rate products based on their enjoyment/satisfaction (or lack thereof) of the product. Predicting user ratings is useful because it enables sites like Amazon to estimate what content the user prefers and gives them another asset with which they can tie to the user for other predictive tasks such as purchase prediction.

The Dataset (2)

a) The Dataset

We used the Amazon movie review dataset from Stanford Network Analysis Platform (SNAP), which consists of movie reviews from Amazon. The data span a period of more than 10 years, including about 8 million reviews leading up to October 2012. Reviews include product and user information, ratings, and a plaintext review [1].

Specifically the dataset consists of: product ID, user ID, profile name, helpfulness (number of helpful votes/number of total votes that the review received), rating, timestamp, review summary, plaintext review. We chose to use the (product ID, user ID) pair as the unique identifier for each review. The dataset consists of 7,911,684 reviews with 889,176 unique users. There are 253,059 products, which is not directly equivalent to the number of movies in the set, since a single movie may consist of multiple products such as region-specific releases, format differences, and rereleases.

Since the Dataset was so large, we decided, in the beginning, to use only the first 1.5 million reviews as our dataset. However, we soon discovered that the reviews were sorted by product ID and by using only the first 1.5 million we would be limiting our feature space due to the disparity between the number of products and users. From the 1 million item training set there are only 4789 products for 94,319 users. Obviously, a huge discrepancy. This will also render us unable to predict the ratings of a product based on other ratings of the same product, because we would be taking the entirety of a products review set due to the sorting issue. Instead, we take a random subset of the 7.9 million reviews in order to fill out our set of 1.5 million reviews.

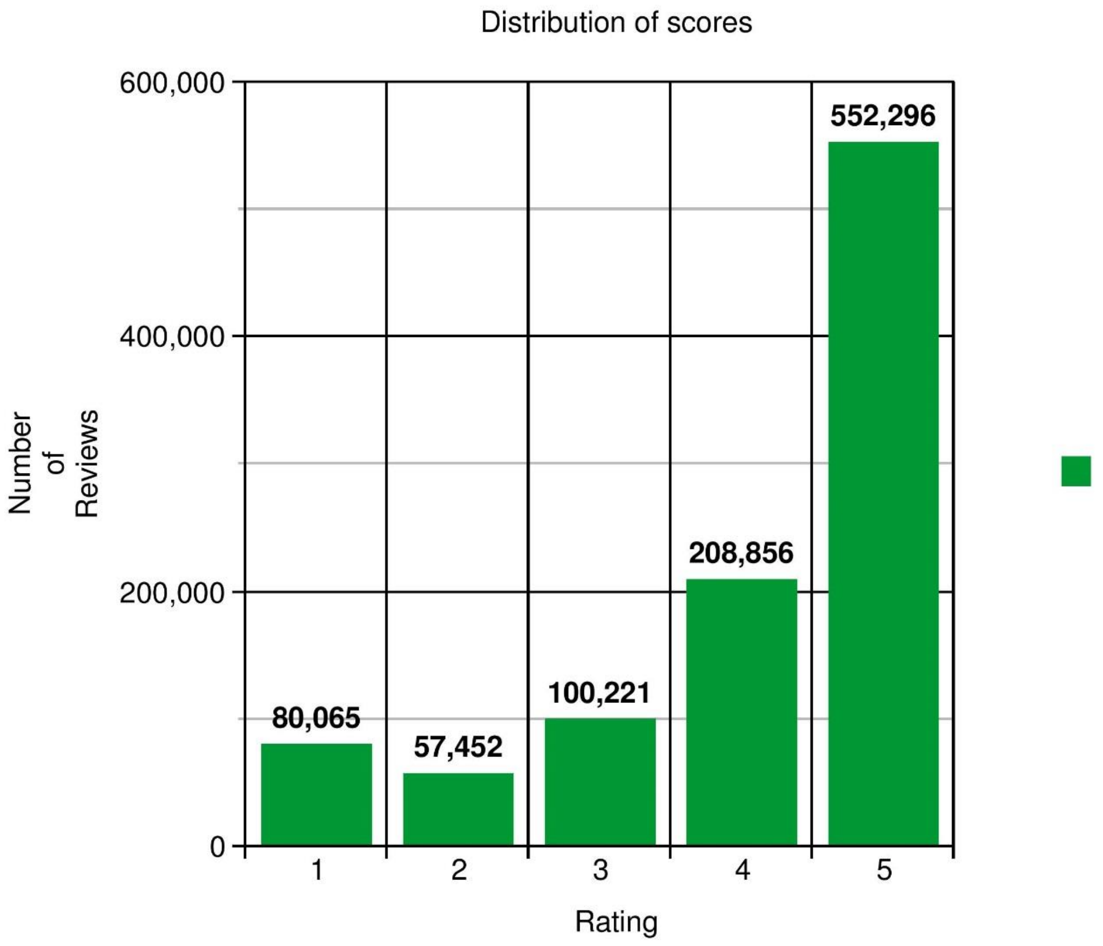
Using this random subset, we would eliminate the deficiency of our previous method. In our dataset now there are 21,721 products and 95,725 users per about one million training set items which is much better than the previously used data set.

We further split our dataset into three random but separate subsets of the 1.5 million reviews where we have 1 million training examples, 100 thousand validation examples, and 200 thousand test examples, so we end up using 1.3 million randomized reviews of the 1.5 million we split from complete set. We will later use these subsets to create a recommender system to predict future ratings of the product based on user.

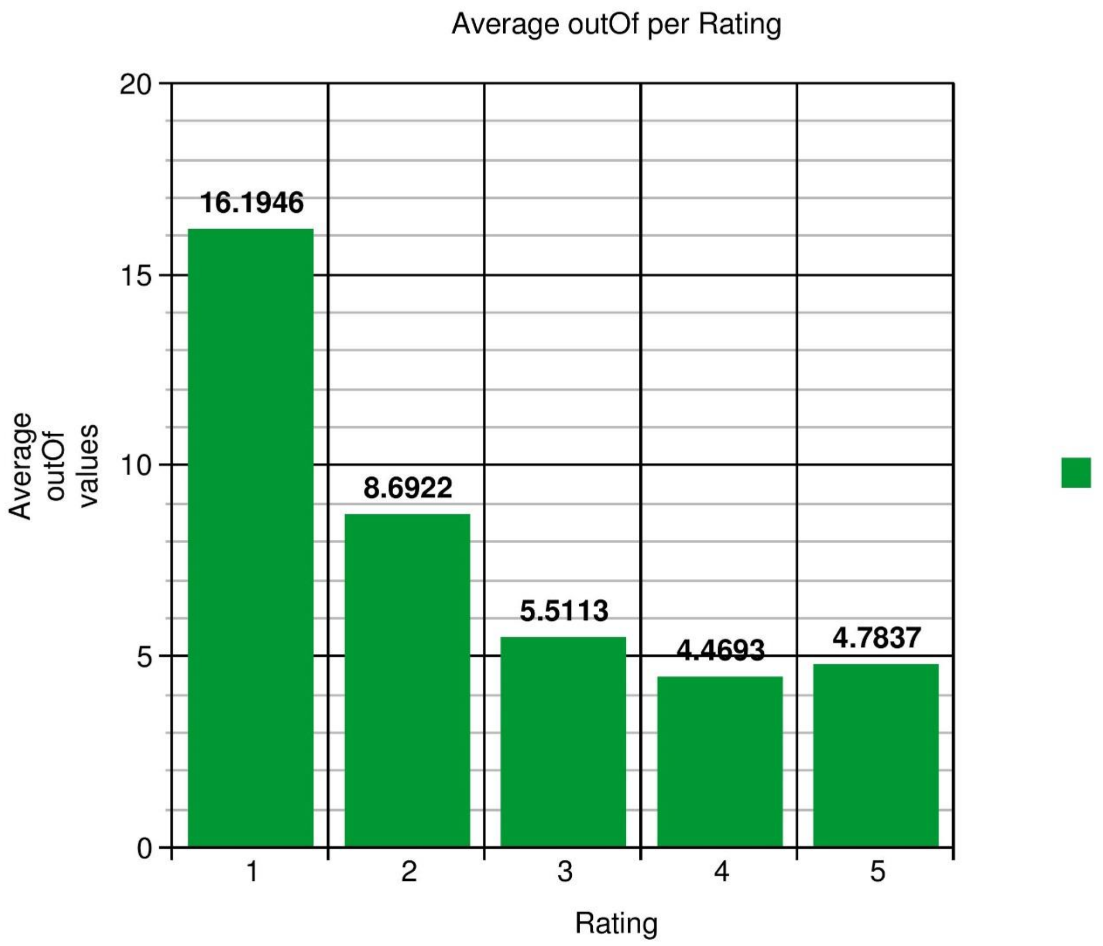
b) Unique Aspects of the Dataset

The dataset has some unique aspects from which we can exploit later and turn into features for some sort of regression.

The distribution of scores, for example, show a heavy bias, unsurprisingly, towards 5-star reviews and less so towards 4-star reviews. This shows that the people that rate products on Amazon tend to only do so if they had a good experience.

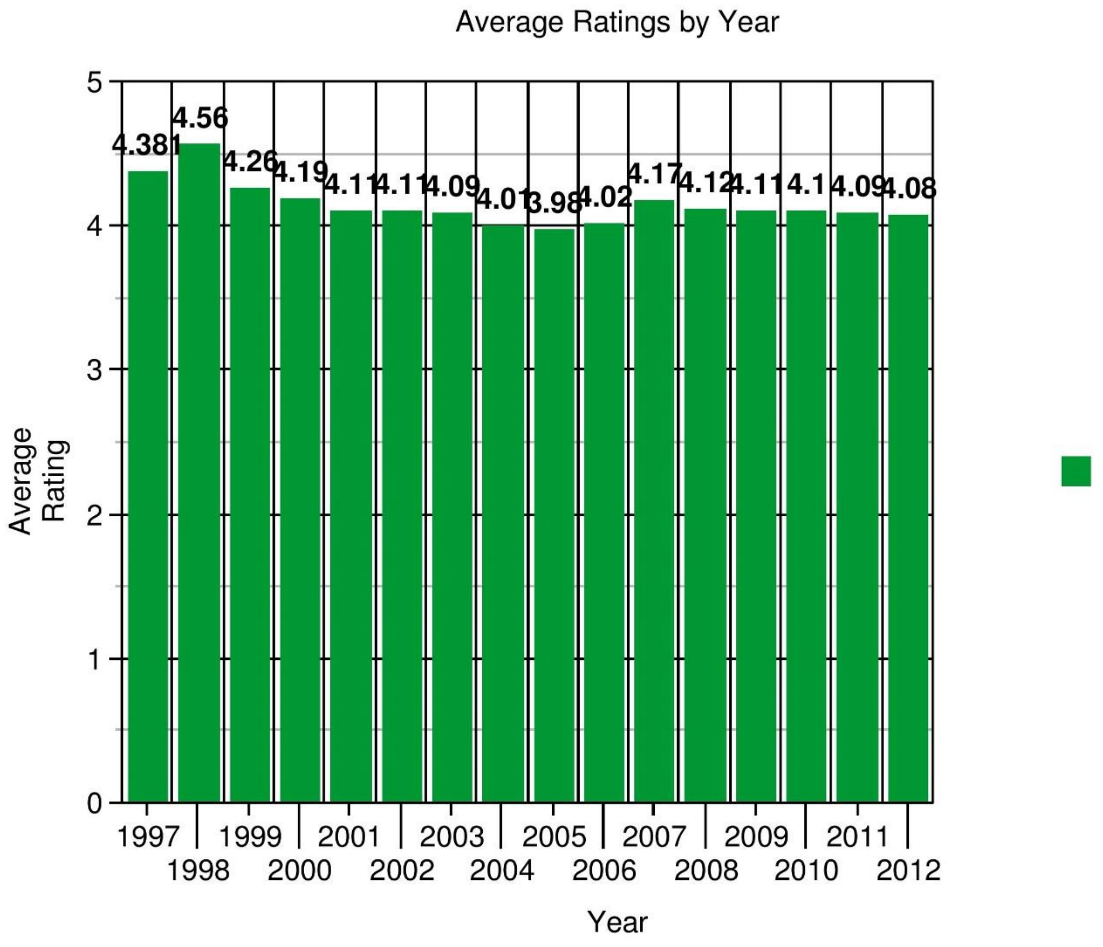


However, the opposite is true for average number of votes for helpfulness. It appears to be that the average number of votes skews much higher for low scoring examples. These reviews could be much more divisive in the community due to the difference between the number of low scores and high scores seen in the earlier distribution. As such, other users might use the helpfulness voting as an agree/disagree button which might explain why the number of votes for the lowest scores are double/quadruple the higher scores.

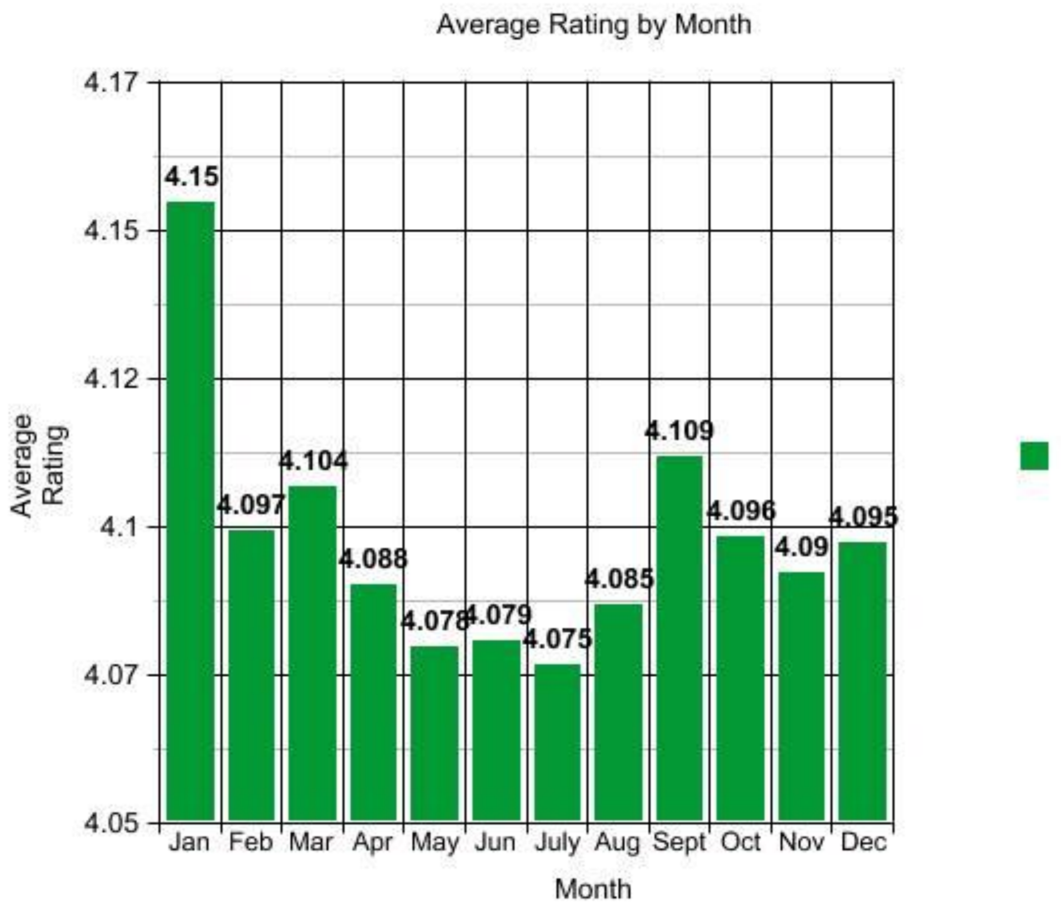


Since the dataset expands over a significant period of time (15 years) and each review has a timestamp, we are able to show how user reviews change over time.

Firstly, we track the average rating per each year of the dataset. We can see that the average review rating takes a dip during the middle years of the dataset, where from 1998 to 2006 we get a steady decline in average review rating. We can infer that there may be a periodic or cyclical element associated with the trend of average review rating per year. Where perhaps in a decade-long cycle movies will get worse and worse until there may be a spectacular year in movies which resets the trend.

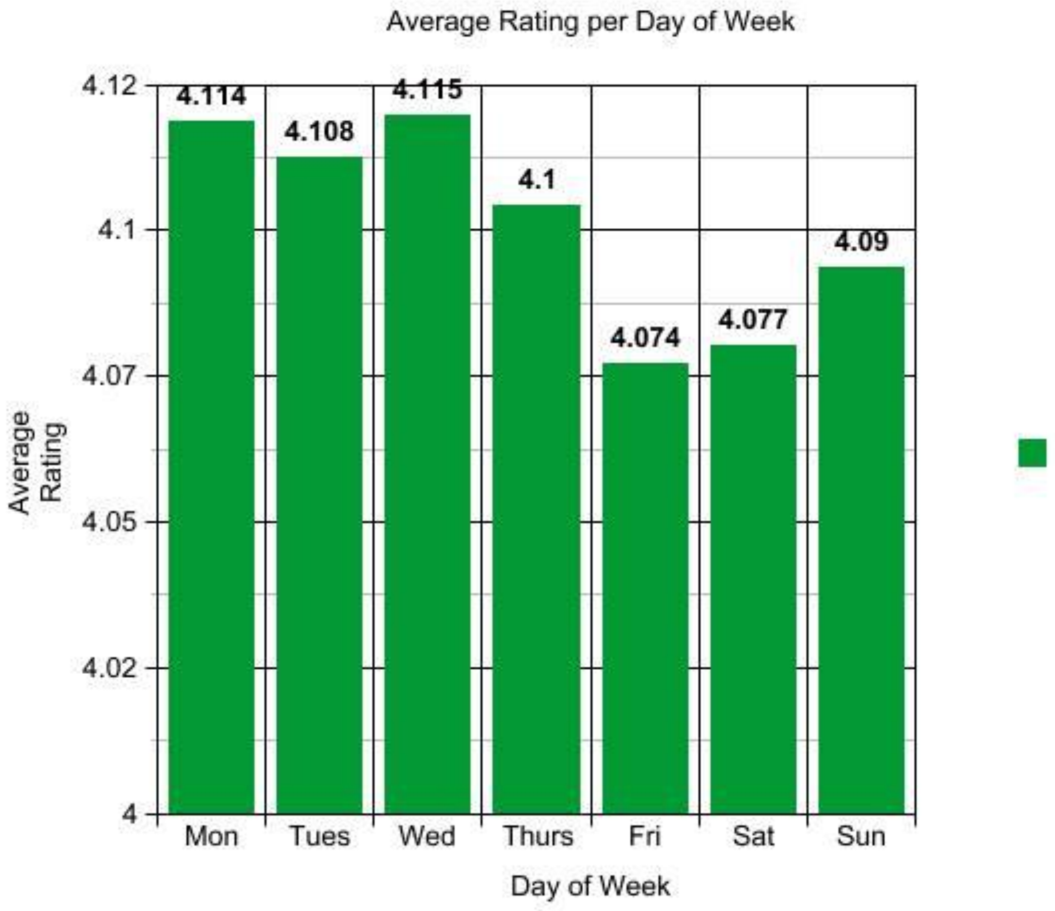


We are also able to show how ratings change over a time period, in this graph we separate the review into monthly bins (reviews go into bins corresponding to the month of their timestamp).



From this we can see than reviews tend to spike in generosity during the winter months and drop during the beginning of summer months and then spike again during September. This suggests that somehow, people are able to enjoy movies more in the winter than in the summer (maybe people like to watch movies together during the holiday season, which hypes up their movie experience).

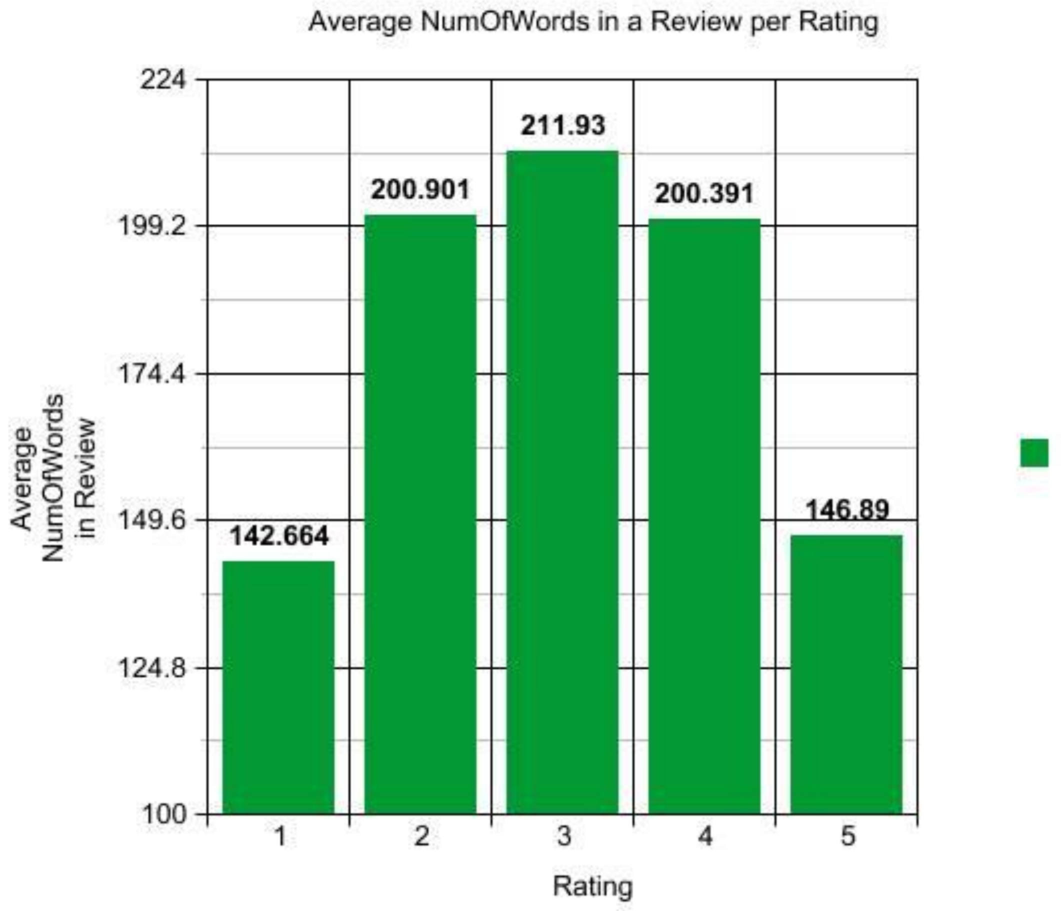
We then attempt to find other time-based periodic trends associated with our data. We bin the reviews into days of the week.



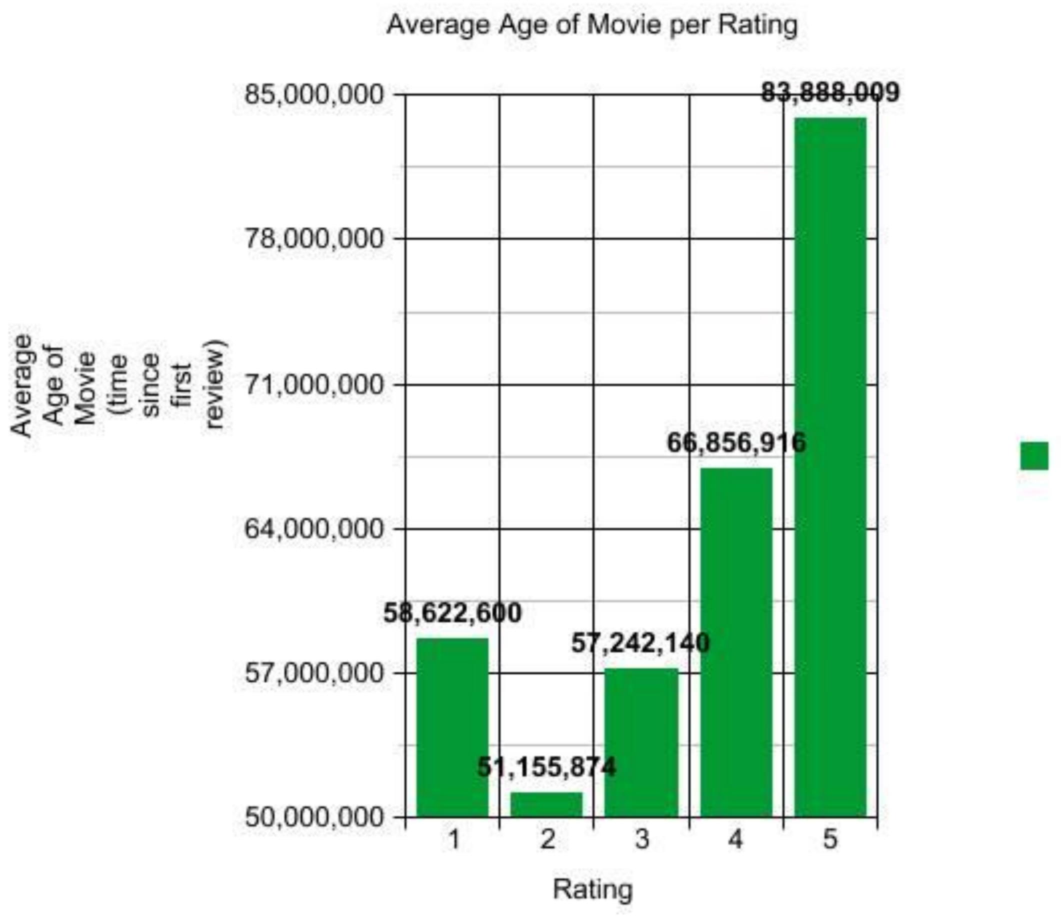
Surprisingly, movies tend to be better reviewed during the weekdays versus the weekends. Which actually may not be that surprising because there is generally more time to write a review during the weekend versus the weekdays.

We can see this could be a factor in our next histogram. For this one, we calculate the average review text length over each rating.

From this we can see that the average number of words is skewed towards the ends, where extremely positive and extremely negative reviews are just two-thirds of the size of the reviews with mediocre ratings. This could be an indicator of more well-thought out reviews being written where the user finds faults in the movie and writes a more nuanced review. Rather than very good movies and very bad movies where users will mainly write about why the movie is so good or so bad, users might spend more time writing about the pros and cons of the movie when they find that the movie is mediocre. However, this feature seems difficult to incorporate into a linear regression model because of its shape.



We can also see that there might be a “classic movie” effect for movies reviewed in the Amazon dataset, where older movies tend to have higher ratings than newer movies. We calculated this by finding the time of the first review written for a particular movie, and subtracting all the other review’s times by the time of the first review to give us an indication of how long the movie has been out for. The distribution makes intuitive sense, again because it might be considered as a classic but also because generally people will likely not bother to watch an old movie unless they hear that it is very good.



<u>The Predictive Task (3)</u>

As discussed earlier, we will be trying to use various regression techniques to predict future ratings for some test set of users for movies in the data set.

a) Assessing Validity of Predictions
If we make a prediction based on the data set we created, how can we be sure that the predictions we make will have any relevance? For us to be sure that our model shows any improvements over a naïve analysis of the data set, we must first do the naïve analysis of the data.

We create a baseline model based on the most relevant feature: the rating. We use the average review rating of the user if they exist in the training set, else we use the average rating of the entire training set. Using this model, we can see that the mean squared error is 1.088, while the mean absolute error is 0.681 on the training set and the R^2 value is 0.681. We use the baseline model as an indicator of whether our models succeed in predicting user ratings.

In order to test our rating predictor we first train on our ~1 million randomized review training set, validate on 100,000 reviews, and test on

200,000 reviews. Each model we use is tested on the 200,000 randomized review items.

We choose to evaluate the models using the mean-squared error, mean-absolute error, and the R^2 value on the test set. We use mean-squared error as our main indicator of correctness, because values that are further away from the true rating are orders of magnitude worse due to the nature of a 5-star rating system. We use the mean-absolute error as a sanity check for ourselves to see the average distance from the true rating each model predicts. We use R^2 (coefficient of determination) because it gives us an objective idea of how well our predictor is performing because the R^2 value compares our predictor to the trivial predictor.

For the predictive task we tried multiple models. We first create a linear regression model without text mining features, then with text mining, and finally we try a multiclass predictor. We use the linear predictor because most naturally lends itself to predict a value, in our case the rating. It also allows us to try multiple different combinations of features until we find a set of features that works relatively well. We then move on to a multiclass model because the value we are predicting lends itself to be split up into different classes because the rating is truly just 5 different star values, so we can easily split each increment of star rating into its own class.

b) Feature Selection

For this task we used multiple different combinations of features to select an efficient linear regression models. For this reason we try different types of features. Since the data set provides no easy way for us to separate the different types of movies into different categories or genres, we are unable to create meaningful features based on these movie distinctions. We can, however, attempt to make a different split based on the review’s helpfulness rating and other discrete features of the set such as review length, average prior ratings of the user, and average rating of the product. These features are part of the data set and no inferences need to be made based on other predictions. However, we do need to preprocess the dataset in order to use these features, such as the average prior rating of the user, where we would need to create lists for each user that corresponded to the time, rating, and product that they reviewed. This will form our basic linear regression model.

We then try basic semantic analysis techniques, such as top 1000 unigrams. In order to use these we needed to shrink the training set by an order of magnitude due to long run times associated with finding unigrams.

From our exploratory analysis of the dataset, we can get a general idea of some of the trends of the dataset, as well as what kind of regression models we might possibly use these features in. For example, it is difficult to use the number of words in a review as a feature for linear regression, because it is difficult to map the shape of the distribution into a linear shape. On the other hand, something like helpfulness outOf values might be more suitable for linear regression.

Other aspects that could be binarized could be average age of the movie and the average number of helpfulness votes where there are clear separations between each rating.

In our models we ended up binarizing all of the features we used except average rating that user gives and the average rating that the product receives.

Models (4)

For this task we first chose to use multiple, different combinations of features to create different regression models.

The various models we tried are:

1) Baseline: We use the user's average ratings if they exist in the training set, otherwise we use the average rating for the entire training set.

2) Linear Regression: We tried to optimize a linear regression model by testing out various features, most of which we observed in our dataset analysis. We tried using the number of helpfulness votes (both up and down-votes), review length, age of the product from the first review in the training set (how "classic" it is), the user's average rating from the training set (or the average rating of the entire training set if the user is not in the training set), how experienced the user is (based on how many reviews the user has written), and temporal factors such as rating tendencies on a certain day or month as features. We later optimize the model by adding unigram features.

3) Ridge Regression: We use the same features as in linear regression but we change the lambda values.

4) Multiclass Classification: We will use the same features that we use for linear regression when fitting our multiclass classification model.

5) Support Vector Machines: We use the same features as linear regression.

We decided to start with linear regression because that is the technique that we were most familiar with from class and its speed in comparison to the other models we tried. We try multiple different linear regression models at first without text features. These models either performed worse than baseline or about the same. The problem with not using text features, is the lack of strong features in the give data without resolving to use real text features due to the simplicity of the discrete data given in the user reviews items.

We then tried linear regression with text features in order to optimize the model. We try to use a unigram model of the top 1000 words, but we cannot realistically run this on the full training set. We decided to, again, shrink the training set to a 100,000 review randomized subset of the 1 million we had before. We use this smaller training set going forward since we include the unigram model each time. This model seemed to have the best performance in comparison to the baseline.

We tried other models too, but they had various issues which caused us to abandon them. We tried ridge regression first due to its relation to standard linear regression and it is still relatively fast. We try multiple values of lambda, but the improvements seem miniscule.

We then moved on to a one vs rest multiclass classification because of the suggestion of the professor. This classification model trains a classifier for each class which in our case means that each star rating corresponds to a class. It seemed, however, to have much higher error when compared to the linear regression with unigram model even though they both used the same features. We think this may be because we had to use a smaller dataset to train the classifier due to the unigram text features. Because of this, we ended up not having enough data to train multiple classifiers.

Lastly, we tried to use a Support Vector Machine to train a classifier using the same features as before. This model proved to be too computationally complex when we used a training set of 50,000 so we moved to a set of 10,000 training examples. Even this seemed to be too complex and we were unable to run it at testable speeds so we abandoned this model.

Related Literature (5)

As discussed earlier, this dataset comes from the SNAP large network dataset collection. The Amazon movie review dataset is part of a set of product reviews that include beer, wine, fine food reviews, as well as being a subset of a larger Amazon product review dataset. The data was used as part of a recommender system analysis wherein the authors compared different models of recommender systems.

The main model created in the research by McAuley and Leskovec was based on the “expertise” of the user [2]. The authors leveraged the temporal aspects of the user reviews to create a timeline for each user. They suggest that the user gains “experience” as they review more items in the dataset, and they infer from this that each user goes through different phases of expertise at their own pace. By using the different phases as a guide, the authors create different latent factor models corresponding to the differing levels of “expertise” from which the user moves from model to model as they become more experienced. Some of the features that we use (user expertise and movie age) are similar to what the authors discuss in their research, though in a much simpler form, and we found that the findings of the authors seemed to match the results of our model as well.

For this dataset, we have seen other datasets. The most recent in relation to this class is the dataset used in Assignment 1, where we created a helpfulness predictor based on the Amazon book reviews dataset. In the provided dataset we were given all the data elements in the Amazon movie dataset we used in this paper along with categorical elements corresponding to the type of book. This would have been helpful in separating the movie data into different types or genres and generating average ratings associated with each genre, since these may differ.

Other versions of the Amazon data set also include more aspects of the Amazon user profile into each user review item, such as gender of the user or helpfulness rank of the profile. The helpfulness rank, for example, is very useful in predicting the helpfulness vote of the user's review.

Many papers we viewed seemed to use some variation of the collaborative filtering technique versus the linear regression model that we created [3, 4, 5, 6, 7]. Collaborative filtering is a technique in which we predict items which have not yet been rated by the user based on the ratings of other users with similar tastes (meaning they have viewed some of the same movies and gave them similar ratings).

By using collaborative filtering in conjunction with other methods, these papers managed to make improvements to the standard collaborative filtering model, such as using fuzzy system logic in conjunction with collaborative filtering [6]. However, this does not seem to be very useful in our predictive task since we are not predicting an item which the user has not rated yet; rather, we are predicting the score that the user has given based on other features of the review.

One of the more exotic techniques we see being used is a combination of Linear Discriminant Analysis (LDA) and ordinal regression (regression by ranking on discrete values) [7].

We ended up using the feature-based linear regression model because that was the type of model with which we were most familiar. This model also seemed to lend itself well to a sort of plug-and-play methodology with which we could replace and/or add features as we saw fit in order to find the best model we could create. We also ended up using multi-class classification, which seems to be used in many predictive tasks similar to ours (such as classifying credit ratings) by classifying reviews into a star rating [8]. However compared to the relatively good rating prediction of the papers we read that used more complex models, our rating prediction seemed to encountered more issues and did not improve upon the baseline in any significant way when we tried to use more computationally intensive models such as multiclass classification.

Results and Conclusions (6)

After trying out many features/combinations of features, it became apparent what kind of features worked best.

Model	**Features**	**Mean-Squared Error**	**Mean-Absolute Error**	**R-Value**
Baseline	Average Review for User, or average review of entire training set	1.088	0.681	0.68
	User Experience,	0.9521693	0.69195299	0.59616938

Linear Regression	Helpfulness Votes, Day of Week Review is given, Age of Movie, Average Rating for movie, Average Rating that user gives			
Linear Regression with Unigram Model	User Experience, Helpfulness Votes, Day of Week Review is given, Age of Movie, Average Rating for movie, Average Rating that user gives, Unigram of top 1000 words	0.79500211	0.63920953	0.49776416
Ridge Regression (lambda = 0.0001)	User Experience, Helpfulness Votes, Day of Week Review is given, Age of Movie, Average Rating for movie, Average Rating that user gives, Unigram of top 1000 words	0.79500208	0.6392097	0.49776414
Ridge Regression (lambda = 1000)	User Experience, Helpfulness Votes, Day of Week Review is given, Age of Movie, Average Rating for movie, Average Rating that user gives,	0.79975846	0.6392786	0.50074219

	Unigram of top 1000 words			
Multi-class Classification (OneVsAll Classifier, LinearSVC)	User Experience, Helpfulness Votes, Day of Week Review is given, Age of Movie, Average Rating for movie, Average Rating that user gives, Unigram of top 1000 words	1.23370398646	0.574441530853	0.772442761763
Support Vector Machine	User Experience, Helpfulness Votes, Day of Week Review is given, Age of Movie, Average Rating for movie, Average Rating that user gives, Unigram of top 1000 words	1.11229793873	0.610648255717	0.696428398647

The features that we ended up using in our models resulted from prototyping quickly with the linear regression model. The final features we used are user experience, which we modeled by binarizing the number of reviews (5 bins) that the user has made, the helpfulness vote count which we binarized into 5 bins, the day of the week that the review was made, which we obtained by using a datetime library on the time, then binarizing it into 7 bins for each day. We also used the age of the movie, which we modeled by subtracting all movie times by the time of the first review made of that movie (this gives a relative sense of how old the movie is), which we also binarized into 5 bins which we picked based on the graph we generated when analyzing the dataset. Finally, we had the average rating that the movie received, and the average rating that the user gave, replaced by the average rating of the global set if the user or movie is not present in the training set (these are the only fields we did not binarized). We end up having 24 features because of binarization not including unigrams.

In comparison the features that did not work well were the ones that were not monotonically increasing across the dataset, such as review based on the month. We attempted to use season periodic trends based on this data but it turned out to be rather fruitless. We also attempted to use review length and review by year but these did not improve our errors at all. We tried bigrams but they ended up performing worse and the feature space increased too much resulting in an overly complex feature set and long run times.

The features in our final feature set seemed to work well because these were the ones that were mostly monotonically increasing or decreasing and correlated with rating. Because of this the relationship between these features and the rating were easy to represent.

Beating the baseline proved harder than expected. The distribution of ratings is very biased in that most people tend to give ratings of 4 or 5. Since we use the average rating that people give as the baseline (and the global average when the user is not present), this turns out to be a fairly decent predictor, since the data is very clustered around the average, meaning the value is likely to be fairly close to the average.

Our best performing model turned out to be our optimization of the simplest linear regression model by using text features in a unigram model of the top 1000 words. This seemed to be the best model because it's simpler and it does not overfit the training set as much as the more complex models.

We assume the models that were not standard linear regression performed worse because we had to use smaller training set sizes for more complex models which resulted in not enough data being used to be able to effectively predict the rating.

References

[1] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. (June 2014). Retrieved December 2015 from http://snap.stanford.edu/data
[2] J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In Proceedings of the 22nd international conference on World Wide Web, pages 897–908. International World Wide Web Conferences Steering Committee, 2013.
[3] Guoyong Cai; Rui Lv; Hao Wu; Xia Hu, "An Improved Collaborative Method for Recommendation and Rating Prediction," in Data Mining Workshop (ICDMW), 2014 IEEE International Conference on , vol., no., pp.781-788, 14-14 Dec. 2014 doi: 10.1109/ICDMW.2014.60, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7022674&isnumber=7022545
[4] Taeryong Jeon; Jaewoo Cho; Soojin Lee; Gyeongdong Baek; Sungshin Kim, "A movie rating prediction system of user propensity analysis based on collaborative filtering and fuzzy system," in Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on , vol., no., pp.507-511, 20-24 Aug. 2009 doi: 10.1109/FUZZY.2009.5277415, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5277415&isnumber=5276875
[5] Lei Ren; Junzhong Gu; Weiwei Xia, "An item-based collaborative filtering approach based on balanced rating prediction," in Multimedia Technology (ICMT), 2011 International Conference on , vol., no., pp.3405-3408, 26-28 July 2011 doi: 10.1109/ICMT.2011.6002184,URL:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6002184&isnumber=6001647
[6] Fikir, O.B.; Yaz, I.O.; Özyer, T., "A Movie Rating Prediction Algorithm with Collaborative Filtering," in Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on , vol., no., pp.321-325, 9-11 Aug. 2010 doi: 10.1109/ASONAM.2010.64, URL:
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5562751&isnumber=5562740
[7] Kawashima, T.; Ogawa, T.; Haseyama, M., "A rating prediction method for e-commerce application using ordinal regression based on LDA with multi-modal features," in Consumer Electronics (GCCE), 2013 IEEE 2nd Global Conference on , vol., no., pp.260-261, 1-4 Oct. 2013
doi: 10.1109/GCCE.2013.6664818, URL:
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6664818&isnumber=6664739
[8] Ahn, H.; Kim, K., "Corporate Credit Rating using Multiclass Classification Models with order Information," in International Journal of Social, BGehavioral, Educational, Economic, Business and Industrial Engineering vol. 5, 12 Nov. 2011