Applied Supervised Machine Learning in Amazon Rating Prediction

Team 11, Team members: Ang Li, Chen Wang, Jialing Zheng, Yaowei Zong

Introduction

Ratings and reviews have been an important part of the e-commerce website for a long time. Not only ratings and reviews eliminate the possible doubts of the customers, but they also increase the likelihood of purchase by 63% [1]. However, it might be confusing for the customers to look directly at the ratings and reviews because the definition of "four star" and "good" differs by the reviewers. For example, a customer may rate on items in a very casual way. Although he thinks that the item is not very satisfactory, he might still give it 4 stars. He thinks his review reasonable because the lacking 1 star means "this item has its weakness, not very satisfying". However, if another user is relatively careful about making reviews on the item, he always gives three star to items which is not bad, four star to those surprise him a little bit, and five star to what he thinks is a real bargain. When he looks at the rating of an item he plans to purchase, he does not know whether other users bought this item and gave rating happens to be generous or careful. Thus, this kind of confusion might prevent the customers from finalizing their purchases.

To eliminate this kind of confusion, make the ratings and reviews better benefit the customers, and increase the revenue of the e-commerce website owner, our team proposed to provide customers with personalized predicted rating, which is a prediction of the rating the customer will give if they eventually purchase the product. With personalized predicted ratings, the customers will stop wondering about how satisfied they will be with the product and know this product will give them for example a "four star" satisfaction on their definition. Furthermore, for Amazon itself, or other e-commerce websites, the prediction of users' rating on a certain item can help to improve the recommendation system. It can be a criterion to decide the priority of items recommended by other recommendation algorithm (such as browsing history based or similarity based system), to make sure that users not just like these items but may probably buy them, considering the aspect like price.

The method we used to predict what rating the user will give on specific items is through machine learning. We have tried linear regression, ridge regression, decision tree and genetic algorithm for this five level classification problem. We used the following five attributes as features: number of reviews of the item, average rating of the item, quality of the item, price of the item, and the user's habit on rating. The rating that user made on that item was treated as label.

To obtain both training dataset and validation, we randomly divided the original dataset into two datasets with respectively 70% and 30% of the original data (with no overlap). We evaluated our prediction's correctness by computing several values: Root Mean Square Error (RMSE), Mean Absolute Deviation (MAD), accuracy, and difference distribution.

Problem characterization

We obtained the datasets of Amazon reviews and metadata spanning from May 1996 to July 2014 from Assistant Professor Julian McAuley's Amazon product data page at University of California San Diego [2]. There are multiple datasets provided on that page and we decided to use the dataset only including users and items having at least five reviews and the dataset that contains the metadata of the products.

The review dataset contains about 41.13 million reviews and provides attributes including ID of the reviewer, ID of the product, helpfulness rating of the review, text of the review, rating, summary of the review, etc. The metadata dataset contains metadata of about 9.4 million products. It provides attributes including ID of the product, price in US dollars, sales rank information, list of categories the product belongs to, etc. With these datasets and data fields therein, how to utilize them is a big challenge for us.

First, we should make sure it contains enough data for us to perform machine learning on it. For example, we should make sure for each user, the number of reviews he gives reaches certain magnitude. Likewise, for each item, the reviews be given on it should also be sufficient, and throw away those invalid data.

Second, we should notice that range of value in different data fields varies greatly. We need to make sure when we do the evaluation (similarity measuring), one particular data will not be totally dominant the resultant value of evaluation. Also, we need to consider the problem of data noise when processing the data.

Furthermore, if we need to predict what rating the user will make on specific items through machine learning, two variables are very important: habits of the customers and quality of the products. However, these two important data fields are not provided in the datasets. It means we need to perform some computations based on the raw data to get these two values. When calculating these values, some problems should be considered. For example, when trying to figure out the habit for a customer, we should consider the customer's reviews by categories, because customers may use different judging standard when it comes to products of different categories. It is possible that one's reviews may be very strict in the categories like clothing, but very generous in categories like snacks. But it is hard (or illegal) to get the customer's personal (maybe too sensitive) information which might potentially lead to different rating to the item. Moreover, "category" itself actually could not be quantified as a feature of input of machine learning. Thus, we need to figure out some other way to deal with this category problem.

Another difficulty is that when we try to get the real quality of a product, we do not get the real data of quality of the product, like "maximum heat (°F) of oven mitts". What we can do is to use the large volume of reviews to get an estimate of quality of the product. Thus, how to exclude factors like personal tastes and customer habits when using reviews given by people who has already purchased it to get quality of products needs to be considered very carefully.

Dominant approaches to the problem

In order to generate accurate predictions and ensure real-time requirements of the prediction(recommendation) system, the researchers proposed a variety of approaches to the problem, such as using Latent Factor Model(LFM) [3], Collaborative Filtering(CF) [4], natural language processing on review texts with Novel Learning Framework [5] and Convolutional Neural Network(CNN) [6]. We present brief descriptions for these approaches in the following.

Latent Factor Model uses the existing available ratings to form a model to perform predictions. This model can be applied on a scale of millions of reviews. Stochastic gradient descent was used in this model by Graff and his team to calculate the predicted rating per user-product pair [3]. This model has the advantage of easy performing and scalable. However, the LFM may not work well on inherent sparse data. It is not accurate when applied with limited numbers of factors and may produce worse result with inappropriate filtering. Overall, this approach is highly relied on the quality of the data and choice of factors.

Collaborative Filtering is another popular approach for recommendation prediction. In practice, CF predicting is based on user profiles that is either user based or item based. This approach has its limitation to produce predictions on new users with few ratings and new item with few ratings. and hard to predict users with unique preferences. To alleviate those issues, another approach working on review texts comes to play. Instead of processing the product rating itself, researchers use natural language processing on review texts using Novel Learning Framework [5] or Convolutional Neural Network [6].

Review texts give much richer information compared to rating numbers and they are able to derive user's preference from text. This solve the "Cold Start" problem and has significant improve over other approaches on unpopular products and inactive reviewers. However, natural language processing is much more complex and time consuming than other models.

Your Methodology

To analyze user's habit on rating Amazon items, quality of each item, and to predict the rating the user will give, we extracted the item ID, user ID, overall rating from each of the reviews of one specific category and the item ID and price of the item from the product metadata. Then, in order to analyze user's habit on rating Amazon items, we aggregated all the reviews by user ID, and used the aggregated reviews to calculate average rating of each user. In order to analyze the quality of each item, we first calculated the global average rating of the category to come up with a standard that what a normal Amazon customer's average rating should be. Then we scaled each of reviews using the following method:

$$Customer\ Adjusted\ Rating = \frac{Customer\ Original\ Rating * Amazon\ Category\ Average\ Rating}{User\ Average\ Rating}$$

We adopted this adjustment method to make sure that the average of adjusted rating of each customer will be exactly the Amazon category average rating. In this way, we equalized all the customers' standard on rating the item and coped with the case that some of the customers are

stricter than others and some of the customers just give five-star ratings all the time. Then we calculated the average adjusted rating of each item based on the adjusted ratings we just calculated. Because personal habits on rating item are removed from the adjusted ratings, we regard the average adjusted rating of each item as its quality. In addition, we calculated the number of reviews of each item to experiment whether it has connection to user's final ratings. We also calculated the original average rating of each item because it is displayed on Amazon product page and might affect customer's expectation of the item and thus affecting the final rating the customer gives. Now for each customer item pair, we have the number of reviews of the item, average rating of the item, quality of the item, price of the item, and the user's habit on rating (average rating given by the user), we're ready to perform machine learning.

Then we performed several machine learning techniques to predict what rating user will make on specific items.

The first technique we performed is genetic algorithm. Genetic algorithm is an algorithm with very long history, it was first used in 1950s [7]. The algorithm mimics the evolution of the genetic world. In genetic world, when producing new children, two chromosomes perform crossover and mutation to ensure the children is not fully the same as parents. Then based on the rule of "survival of the fittest", the chromosomes of best fit are more likely to be passed down to next generation (by not being killed by predators or preferred by opposite sex when finding partners). We're performing similar techniques in the genetic algorithm. First, we determined that we want to start from the most basic way, linear way, of predicting user ratings. So, we came up with the following formula for prediction:

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}, \qquad i = 1, \ldots, n$$

In the formula, p is the final prediction we make, f_i is the value for the i^{th} feature, c_i is the coefficient we want to generate by using genetic algorithm, b is the bias we also want to generate.

In experiment, we first randomly generated the array $[c_1, c_2, c_3, c_4, c_5, b]$ for each "chromosome" in the population pool. Then we calculated the fitness for each "chromosome" by using them to predict the user ratings and find out the Mean Absolute Deviation. The fitness was defined by $1/(MAD)^3$, this is to ensure that "chromosomes" with a low MAD will be assigned higher fitness and there is enough reward to "chromosomes" with a relatively low MAD. After that we started to choose "chromosomes" for the next generation. We picked "chromosomes" in a randomly fashion but higher fitness "chromosomes" will have a higher chance of getting selected. Then we randomly perform crossover and mutation on the newly picked "chromosomes". After the crossover and mutation, we've generated the next generation of population pool. We repeatedly performed these techniques until a given number of generation has passed. We perform calculation for fitness for one last time and pick the "chromosome" with highest fitness as the training result.

The next technique we used is Linear Regression. Linear Regression is one of the most classical model in the machine learning. It can finally generate a hypothesis that match the form:

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i, \qquad i = 1, \ldots, n$$

The core approach of the linear regression is to find the optimized vector beta, thus can provide the minimum error in the model. In the classical statistical, if the error (optimizer) were defined

by least square error, which is

$$RMSE(h) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - h(x_i)\right)^2}$$

Then, the optimized vector beta can be derived by

$$\mathrm{b} = (X'X)^{-1}X'y$$

However, this method cannot be applied on our dataset since the size of the dataset. This mathematics estimation of beta needs to put everything in a single matrix which is impossible for large amount of data. Therefore, we used the Linear Regression with SGD model in the Spark Mllib to train our data. The SGD model would get partial derivation for each feature and then approach the optimization value.

The standard gradient descent algorithm updates the parameters θ of the objective $J(\theta)$as,

$$\theta = \theta - \alpha\nabla_\theta E[J(\theta)]$$

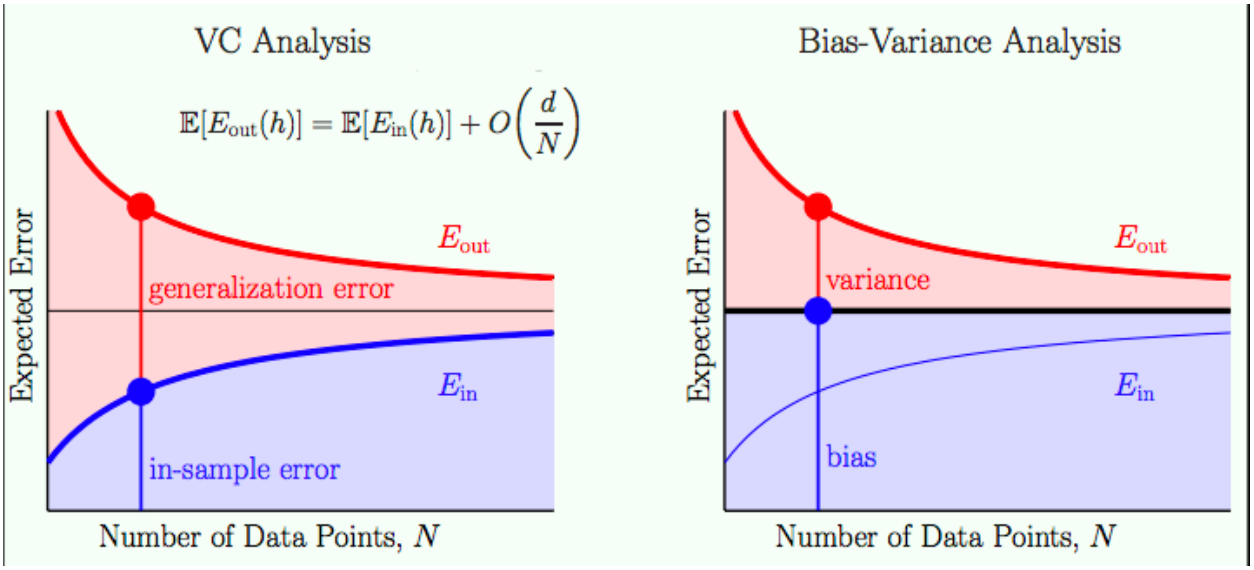
where the expectation in the above equation is approximated by evaluating the cost and gradient over the full training set. Stochastic Gradient Descent (SGD) simply does away with the expectation in the update and computes the gradient of the parameters using only a single or a few training examples. The new update is given by,

$$\theta = \theta - \alpha\nabla_\theta J(\theta; x^{(i)}, y^{(i)})$$

with a pair $(x^{(i)}, y^{(i)})$ from the training set. [7]

Moreover, the SGD can even only use part of the data to perform each training. I.E, 10 percent of the data could be ransomed picked up during each iteration in the training. In this way, the large amount of data can be trained in a limited memory space, computing resources.

We used Ridge Regression as well. Ridge Regression is very similar to the linear regression. The goal of the ridge regression is avoiding overfitting during high number of iterations. It adds a punishment term to the RMSE thus the optimized settle point would be more smoothly, which is a balance of the in-sample error and out-sample error.



Fitting analysis [8]

However, during our experiment, the result didn't show any overfitting, thus the set of the lambda (Regulation term) got the worse results than the linear regression so we set it to 0 at the end.

The last technique we used is decision tree algorithm. We also performed experiments based on decision three model in the Spark Mllib library. From its official documentation, the decision tree in Spark was realized by Random Forests.

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. [9]

And the decision tree has the best results among all other algorithms. It can reach the minimum RMSE, MAD and highest accuracy on 10 nodes training. But the con part of the decision tree is overfitting. I.E The results of training that more than 10 nodes have higher out-sample error than in-sample error which means the number of nodes should be smaller and as well as the branches.

Experimental Benchmarks

In this experiment, we performed four techniques to use as benchmarks, including Mean Absolute Deviation, Root Mean Square Error, Accuracy, and Difference Distribution. In each run, we randomly separated data into a 70% training dataset and a 30% validation dataset (with no overlap). As a result, we were able to produce both the in-sample benchmark and the out of sample benchmark.

First, we performed Mean Absolute Deviation (MAD) which is defined by the following formula:

$$MAD(h) = \frac{1}{N}\sum_{i=1}^{N}|y_i - h(x_i)|$$

The Mean Absolute Deviation is calculating the average of the difference between predicted value and the actual value. It can provide us with some idea what our average error is. We calculated average MAD of three techniques and presented them in the Figure 1. As we can see from the figure, on average the error of our predicted rating is around 0.5.

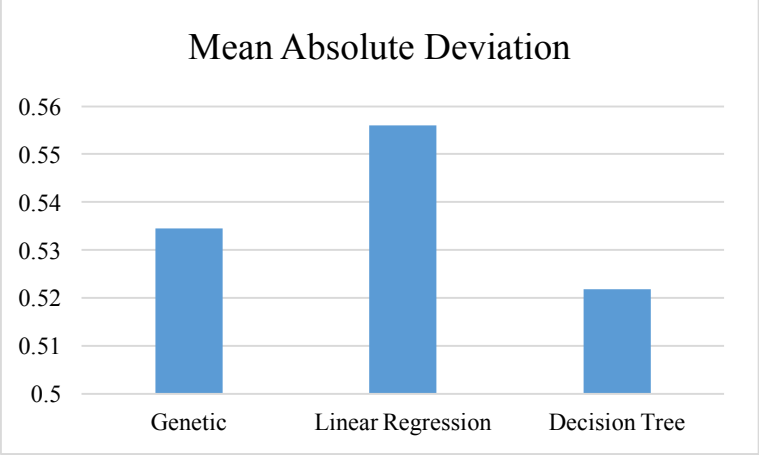


Figure 1. Average Mean Absolute Deviation of Three Techniques

Then we also calculated the Root Mean Square Error which is defined by the following formula:

$$RMSE(h) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - h(x_i)\right)^2}$$

The Root Mean Square Error calculates the square root of the average of the squared difference between the predicted value and the actual value. Because RMSE first performs the average after the error is squared, the puts more weight on those large errors compared to Mean Absolute Deviation [10]. That's the reason why we chose RMSE in addition to MAD as our benchmark. We presented the average RMSE of three techniques in Figure 2. We can infer from the figure that the average root mean square error is between 0.85 and 1 for genetic algorithm, linear regression, and decision tree techniques.

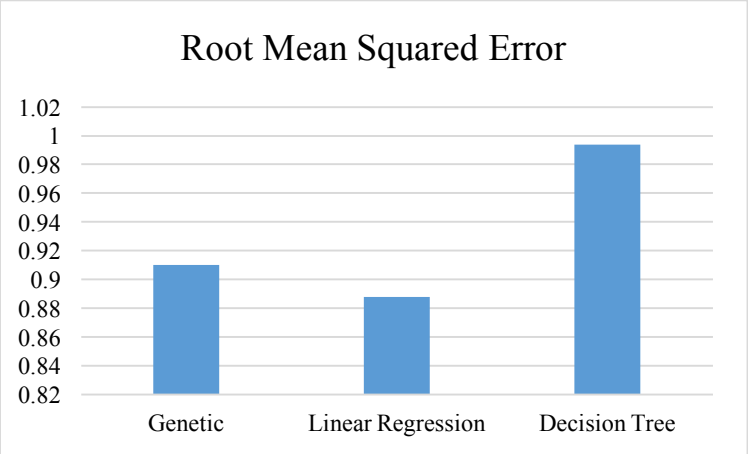


Figure 2. Average Root Mean Square Error

We also analyzed the accuracy of our prediction, that is the percentage of predicted rating that is the exact same as the actual rating. From this benchmark, we will know what is the possibility that when a customer see the predicted rating we produced, the predicted rating is the exact rating he will give when rating. We presented the average accuracy of three techniques below. As presented in the figure 3, decision tree algorithm provides the best average accuracy of a little more than 65%. Genetic algorithm provides the second-best accuracy of around 58%. Linear regression has the worst accuracy of around 55% percent.

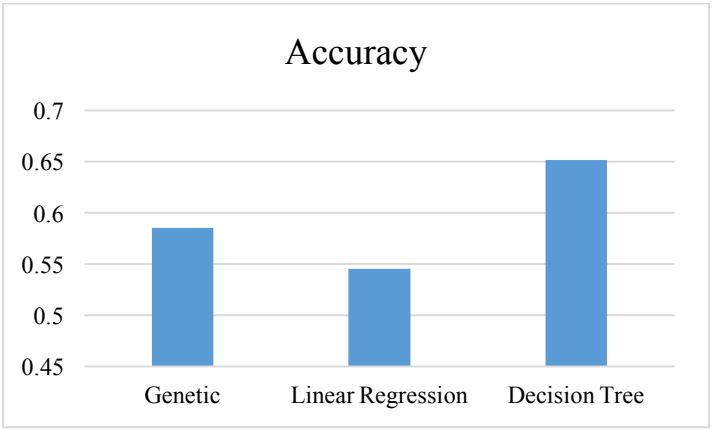


Figure 3. Average Accuracy

In addition to accuracy, we wanted to know when our prediction is not accurate, how far away is our prediction from the actual rating. So, we introduced a benchmark to test the difference distribution. For each prediction we make, we calculate the difference between our prediction and the actual rating the customer gave. This will result in a number between -4 (customer gave rating 1 but we predicted rating 5) and 4 (customer gave rating 5 but we predicted rating 1). Then we analyzed the distribution of the resulted number and produced the following table (only Amazon Instant Video category is presented in this paper). From the table 1, we can see that although decision tree algorithm performs relatively well in accuracy, it also has the largest possibility to perform extremely bad (difference is -4 or 4).

Difference	-4	-3	-2	-1	0	1	2	3	4
Genetic	0.25%	1.96%	3.73%	13.78%	73.77%	6.45%	0.06%	0.00%	0.00%
Linear Regression	0.06%	1.26%	2.47%	11.13%	67.51%	16.75%	0.82%	0.00%	0.00%
Decision Tree	0.51%	1.58%	3.98%	14.16%	74.59%	4.87%	0.13%	0.13%	0.06%

Table 1. Difference Distribution

Insights Gleaned

We have learned a lot from this project and the most important discovers were following:

1. The mode we got from overall products on Amazon is lower than single category.
2. The effect of normalization to linear regression and decision tree.
3. The necessarily of the ridge regression.
4. Decision tree overfitting.
5. The optimization of the genetic algorithm.

We run four algorithms in this project in total, which were genetic algorithm, linear regression, ridge regression and the decision tree. Among all these four algorithms, the decision tree has the best performance, which can reach MAD:0.54, RMSE:0.93 and accuracy: 62.32% on overall datasets and reach MAD:0.32, RMSE:0.74 and accuracy: 76.54% on "Amazon_Instant_Video" category. We thought the reason is because users have different expectations on items from different categories. In that case, the category should be treated as a feature in the training. However, there's not relationship between different categories thus we cannot assign labels like 1, 2, 3, 4 to the items. However, the decision tree can still get the right results even if the category feature is discrete.

Another interesting finding is the normalization of the data. Before the normalization, we cannot get any useful results from the linear/ridge regression, all the results were NAN (Not a number). But the decision tree can still get nearly 80% of the accuracy. We thought the reason is because the linear regression model is using SGD thus, the range of different features affect a lot. The step size of different partial derivative must match all features in the linear mode, otherwise it won't work. As for the decision tree, since every branch could be able to decide which way to go, it doesn't matter if any feature's range is much larger than others.

The third one is about the overfitting. We first thought we could have some overfitting in our linear regression mode so we introduced the ridge regression which is composed by linear model and a punishment term. However, we didn't find any overfitting in our partial training. All the in-sample error match the out-sample error in the linear models. Thus, we thought that we may not need ridge regression any more in this project but we tried any way. Then we got the result worse than the linear mode, so we choose not using ridge regression finally.

Another discovery is about the overfitting is about the decision tree. We used the mode provided by the Spark Mllib and we found that we specify the maximum depth of node to over 10, we can get smaller in sample error but the out-sample error started to increase. Therefore, we finally stop our depth of the node to 10 and then get the best results.

As for the genetic algorithm, we first cannot get results from it since this is totally self-made map-reduced based algorithm. It runs super slow and not stable. We first increased the punishment level for model that has the worse results and then add more mutation rates and then finally get even better results than the linear regression.

How will the problem space transform in the future

Prediction/Recommendation system plays a significant important role in e-commerce market. With the promotion of Netflix Prize competition starting from 2006, researchers have been experimenting with a variety of approaches to generate more accurate predictions. From the winning experience of Koren and Bell [11], who utilized a combination of Latent Factor Models, SVD-based models, and Collaborative Filtering, we can see that it is more effective to use multiple approaches all combined together.

Such improvement in prediction can change our shopping habit. Image with a fully developed recommendation system, people can easily get satisfied products without further investigating by themselves. This will reduce the amount of time people now spending struggling making decisions and dealing with unsatisfied purchases. This feature might be included in personal assistance in the future, like Amazon Alexa, Google Assistance and Siri, and you can ask for advice to purchase anything by simply saying "I need a new computer for work" or "Give me a book recommendation list" and you will get customized recommendation based on your personal profile at once. This could all happen in the Cloud with scalable analytics and machine learning on Big Data.

Furthermore, scalable analytics is not limited to e-commerce alone. With the rapid pace of snackification in every area in modern lifestyle, people will benefit a lot with the help of scalable analytics. Shopping recommendation, earthquake detection, risk assessment, decision making, artificial intelligent, pretty much every field could take advantage of the power of scalable analytics.

From our experience during the experiment, we were confronted with limitations of computing abilities and available memory on the laboratory machines when dealing with large scale of data. We can predict that with more powerful resources the processing time will reduce evidently. We may get better results if we increase the iteration times while training our model during machine learning with some of the algorithm, say Genetic Algorithm.

Large companies such as Amazon, Google, Netflix with powerful resources and advanced technologies are already exploring and delivering the features we discussed above. Take Netflix as an example, they are willing to $1 Million prize to seek substantially improve their accuracy of predictions on how much someone will enjoy a movie based on their movie preferences. We can conclude that they take this feature seriously and the main competition in the future is about how good you can predict the needs of your customers, not only about what you can provide to buy for your customers. With the evolution in computing abilities and data mining, such scalable analytics will play an important role in people's daily life and deliver more valuable services.

Conclusions

From previous insight section, we found that our models get better results on separate category than over all result. Here's a table that about the results from over all results:

Genetic Best			Ridge Regression Best			Decision Tree Best		
MAD	RMSE	Accuracy	MAD	RMSE	Accuracy	MAD	RMSE	Accuracy
0.53622874	0.905368673	57.96%	0.55576078	0.88547152	54.40%	0.519019226	0.98515803	64.89%

Table 2 Results from all Categories

	Genetic Best			Ridge Regression Best			Decision Tree Best		
	MAD	RMSE	Accuracy	MAD	RMSE	Accuracy	MAD	RMSE	Accuracy
Best	0.34702908	0.743878084	73.77%	0.36982249	0.71335527	68.79%	0.282541487	0.73317709	80.90%
Worst	0.65942056	1.123643017	49.98%	0.68142055	1.02007489	46.37%	0.684585819	1.24492675	56.41%

Table 3 Results from Separate Category

From the results above, we can easily observe that the decision tree has the best results among other three algorithms. And it has the 64.89% percentage accuracy among all categories results, and most of the single category results has better results than all categories. And the category that fitted worst is the "Cell_Phones_and_Accessories" category. The reason that it has smaller reviews than other categories.

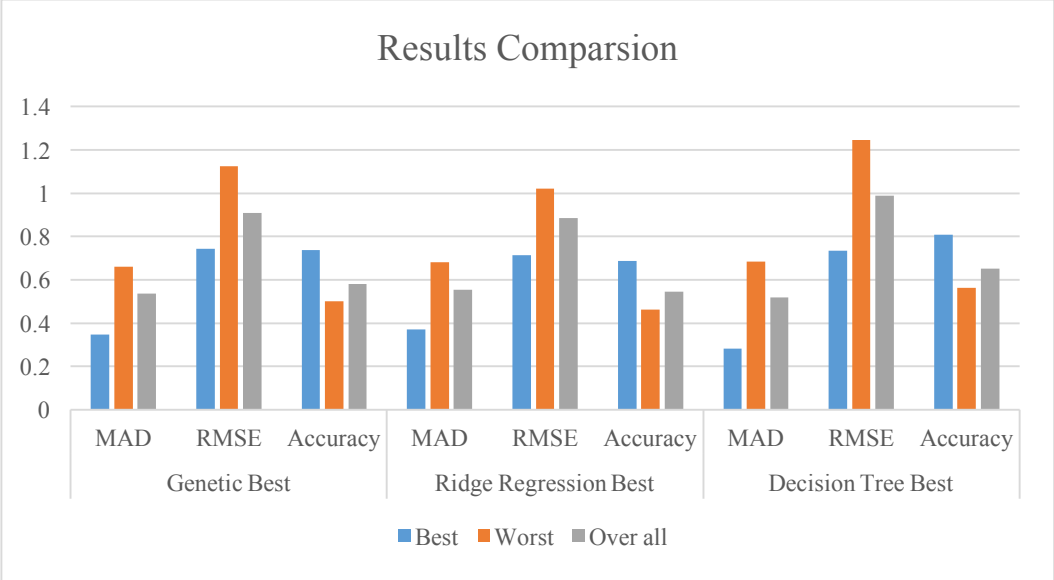


Figure 4 Results from Separate Category

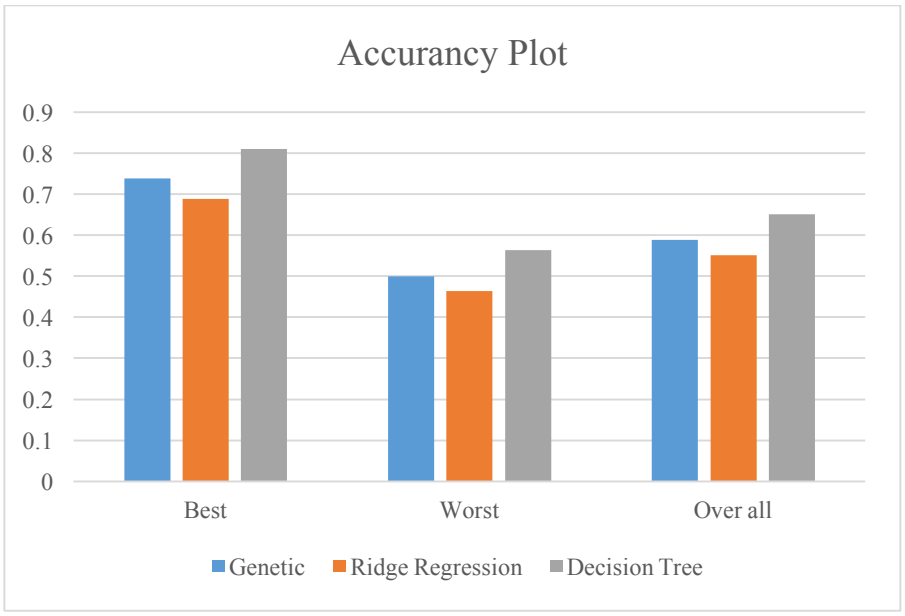


Figure 5 Accuracy Results from Separate Category

It's easy to observe the decision tree has the best accuracy, but all algorithm has similar results, which is nearly 60%, much higher than random rating pick up (20%).
As for the weight vectors,

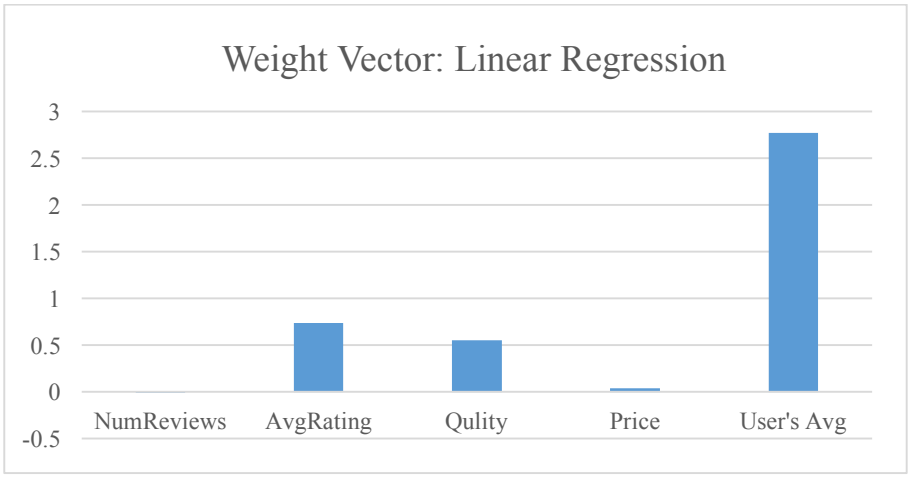


Figure 6 Weight Vector of Linear Regression

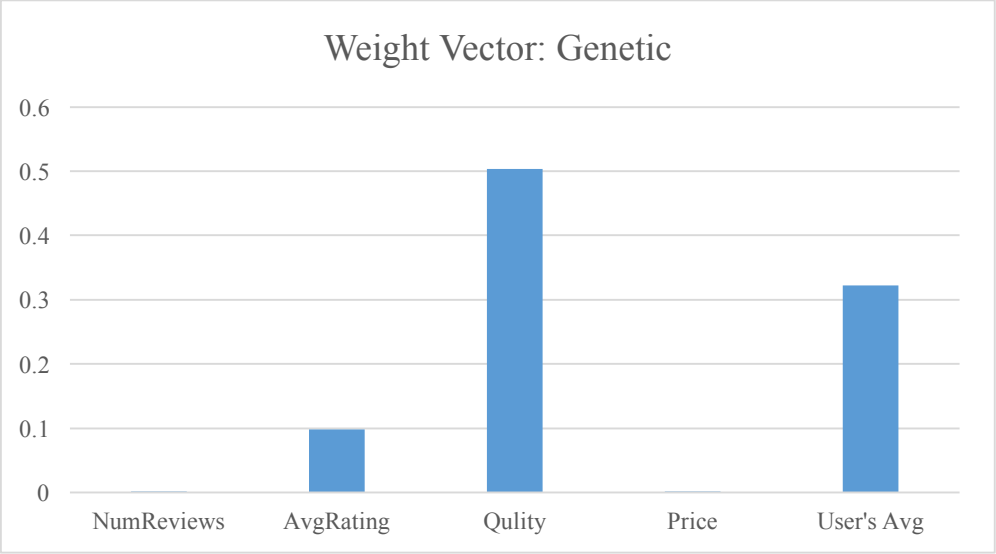


Figure 7 Weight Vector of Genetic

And we can observe that the weight vector of the genetic and linear regression has some differences. But for the noise features such as price and number of reviews, two training mode has nearly same results. Those two training models have very small value of numReviews and Price, which means those two features didn't play an important role in the final hypothesis.

Another interesting part is that the RMSE of Decision Tree is higher even its accuracy is better than other two models. The reason that causes this to happen is because that the decision tree can give out discrete results instead of continues hypothesis like linear regression. Thus, the rounded didn't do anything on results of the decision tree. RMSE represented the results that has more derivation,

Difference	-4	-3	-2	-1	0	1	2	3	4
Genetic	0.25%	1.96%	3.73%	13.78%	73.77%	6.45%	0.06%	0.00%	0.00%
Ridge Regres	0.06%	1.26%	2.47%	11.13%	67.51%	16.75%	0.82%	0.00%	0.00%
Decision Tre	0.51%	1.58%	3.98%	14.16%	74.59%	4.87%	0.13%	0.13%	0.06%

Table 4 Results of accuracy Derivation

We can observe that the linear has smaller error when it reaches to +4 or -4 but it has more error one 1 side. But the Decision tree has more error on further side, which is because it is a discrete classifier.

We found a paper [6], where they have similar experiments done on same Amazon datasets. And we got better results than they get on MSE (Mean Squared Error) aspect.

MSE	Offset	MF	HFT	CNN-only	Attn+CNN	Genetic	Linear	Decision Tree
Amazon instant video	1.273	0.946	0.925	0.944	0.936	0.572	0.509	0.562
Automotive	0.939	0.875	0.975	0.89	0.881	0.678	0.6	0.709
Baby	1.315	1.167	1.151	1.178	1.176	0.977	0.937	1.334
Cds and vinyl	1.15	0.885	0.861	0.875	0.866	0.781	0.775	0.938
Grocery and gourmet food	1.202	1.017	1.004	1.01	1.004	0.846	0.83	1.106
Health and personal care	1.233	1.068	1.053	1.06	1.054	0.903	0.862	1.221
Kindle store	0.916	0.623	0.609	0.621	0.617	0.558	0.544	0.567
Musical instruments	0.742	0.689	0.726	0.71	0.703	0.667	0.585	0.668

Table 5 Results Comparison with Published Paper

We have better results on these categories when consider the MES aspect on all three models. Especially the linear model, which has a continues hypothesis to give the most smoothly model.

Bibliography

[1] Charlton, Graham. "Ecommerce consumer reviews: why you need them and how to use them." *Econsultancy.* Econsultancy.com Limited, 08 July 2015. Web. 05 Apr. 2017.

[2] McAuley, Julian. "Amazon product data." University of California San Diego, n.d. Web. 05 Apr. 2017.

[3] Graff, Casey et al. “Music to Your Ears: Song Recommendation Using Rating Prediction on Amazon Digital Music.” *Semantic Scholar.* 2015. Web. 05 Apr. 2017.

[4] Wang, Jun, Arjen P. De Vries, and Marcel JT Reinders. "Unifying user-based and item-based collaborative filtering approaches by similarity fusion." *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 2006. Web. 15 Apr. 2017.

[5] Li, Fangtao, et al. "Incorporating reviewer and product information for review rating prediction." *Twenty-Second International Joint Conference on Artificial Intelligence.* 2011. Web. 15 Apr. 2017.

[6] Seo, Sungyong, et al. "Representation Learning of Users and Items for Review Rating Prediction Using Attention-based Convolutional Neural Network." *International Workshop on Machine Learning Methods for Recommender Systems.* 2017. Web. 05 Apr. 2017.

[7] "Optimization: Stochastic Gradient Descent." *Unsupervised Feature Learning and Deep Learning Tutorial.* N.p., n.d. Web. 27 Apr. 2017.

[8] http://www.cs.colostate.edu/~cs545/fall16/lib/exe/fetch.php?media=wiki:05_overfitting.pdf

[9] "Random forest." *Wikipedia.* Wikimedia Foundation, 24 Apr. 2017. Web. 27 Apr. 2017.

[10] JJ. "MAE and RMSE - Which Metric is Better?" *Medium.* A Medium Corporation, 23 Mar. 2016. Web. 28 Apr. 2017.

[11] Bell, Robert M., and Yehuda Koren. "Lessons from the Netflix prize challenge." *Acm Sigkdd Explorations Newsletter* 9.2 (2007): 75-79. Web. 26 Apr. 2017.