DSE 220 MACHINE LEARNING

Final project – amazon review helpfulness prediction

Ying Cui
A53292736 | y5cui@eng.ucsd.edu

I. Data Process

1. Exploratory data analysis

The given Amazon review training data set has 2 million review samples and 12 features, and the test dataset consists 14,000 review samples. I started with exploring the data distributions over each given feature. There're 12,5851(63%) samples have no price value, and I dropped the feature. Since our goal is to predict the number of helpful votes which means we need to find the relations between features and review helpfulness. The review samples which have no votes cannot give us reasonable information, so I just visualized the voted-reviews.

- Voted reviews: Nearly 68.5% of the review samples have no vote.

```
[Out]Number of reviews with no vote: 136984
     Number of reviews with vote: 63016
```

Helpfulness: Among the voted reviews more than half of the reviews have 100% helpful rate. [Fig.1]

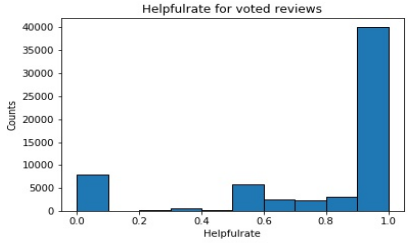


Figure 1:

- Product rating: Fig.2 shows number of voted reviews in each product rating, and fig.3 shows the average helpful rata in each rating. More than half of the reviewed product is got 5 satars and the with rates high product reviews have more votes for helpful which tells us the product rating and helpful rate have linear relations.

Figure 2:

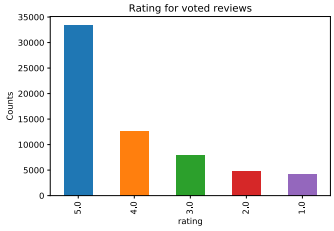
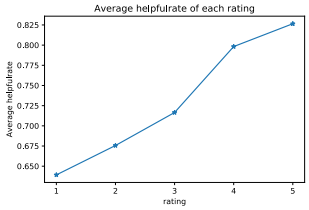


Figure 3:



- Product category: As we can see from the fig.4, most of the reviews in the 0 group and the category and we can't say the average helpful rate and category group have any relations.

Figure 4:

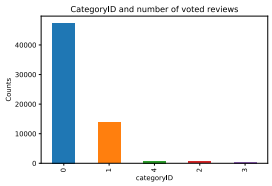
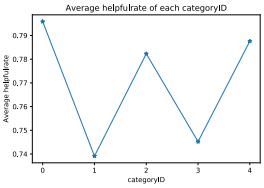


Figure 5:



- Out of (votes): Among the 63,016 voted reviews, most of the reviews have less than 5 votes. Then when I build model, I split the voted reviews for two parts, one for total votes under 5, the other one for reviews votes greater than 5. [Fig.7]

Figure 6:

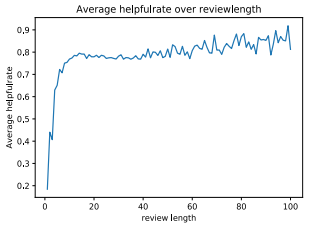


Figure 7:

	range	counts	percentage	avgHelpfulrate
0	(0, 5)	53025	0.8400	0.7652
1	(5, 10)	5322	0.0800	0.8554
2	(10, 15)	1854	0.0300	0.8855
3	(15, 20)	947	0.0200	0.8953
4	(20, 25)	576	0.0100	0.9054
5	(25, 30)	326	0.0100	0.9123
6	(30, 35)	224	0.0000	0.9228
7	(35, 40)	153	0.0000	0.9128
8	(40, 45)	106	0.0000	0.9404
9	(45, 50)	95	0.0000	0.9286
10	(50, 537)	388	0.0062	0.9281

- Review length: Here, review length represents the count of the words in each review. Neary 35% of the reviews word counts in range 10 to 20, and 20% in 20 to 30. The reviews with long text bodies have more helpful votes. [Fig.6]
- Review time: As I observed, the train data reviews is from 2003 to 2014, and most of the reviews is wrote in 2012, 2013 and 2014. There're only 2 reviews wrote on 2003, and every year the average helpful rate reduced a little which 2003 is 0.95 and 2014 is 0.78.

2. Feature selection

After identifying the appropriate features, I further analyzed features using scalar representation and one-hot encoding which can better represent the relations between features.

- Item rating
- Average item rating
- Review time (year)
- Review time range
- Number of votes
- Length of the review & Average review length
- Sentence count
- Character count
- Number of words in all caps
- Number of exclamations

In addition, I captured the unigrams features of each review text using TFIDF ("term-frequency, inverse document frequency"). TFIDF can able to scale down the weighting of stop words and other common words that appear frequently across entire raw data and instead weigh more heavily words that appeared frequently in the single review at hand.

sklearn. feature_extraction.text.TfidfVectorizer(max_features =5000)

3. Data process

All the data saved onto disk in JOSON format with feature name as key and text as value. I created a function that append each feature to a list and append all the list to create X and y value.

II. Model Selection

For testing purpose, I split the train data into 60% for training and 40% for validation dataset. When I add unigram features, I only selected the number of votes greater than a certain number (minOutof) and with a certain maximum features(num_max_feature). I created two distinct models for two dataset partitions, and I trained only voted reviews' data. I tried different ways to split the training data. After tuning on validation dataset, I chose the entries with 1 to 5 votes as a group and greater than 5 for another group. If the out of values equal to 0, I simply predicted as 0 for the reviews with no votes. Since, as we can see from the EDA, more than half of the samples votes is under 5, the way to partition is made sense. Though for the data that number of votes under 6, I didn't included the unigram features, as they didn't fit well with the model. tested various regression models such as SVR , random forest, gradient boosting, ridge regression and elastic-net. I also tested Logistic Regression for those with only one vote, but it didn't perform well on Logistic Regression. For the model evaluation, I used MAE(mean_absolute_error) as a scoring criteria. In the end selected ElasticNetCV and Ridge regression model.

Final solutions:

Training: 120,000 samples; Validation: 80,000 samples; Test: 1,400 samples;

TFIDF: Fit on data with number of votes greater than 20. (minOutOf=20)

```
sklearn.eature_extraction.text.TfidfVectorizer(max_features==5000)
```

- Elastic Net Model (vote=[1,5])
  ```
  sklearn.linear_model.ElasticNetCV(cv=5, l1_ratio=0.8)
  ```

- Ridge Regression Model (votes>5)
  ```
  sklearn.linear_model.Ridge(alpha=5,fit_intercept=True)
  ```

III. Conclusion

I got validation MAE score of 0.58 for first partition, and 3.12 for second partition, combined validation MAE score of 0.31. For the public test score, the best one I got is 0.165.