DSE220 Final

Kaggle user name: w9yan

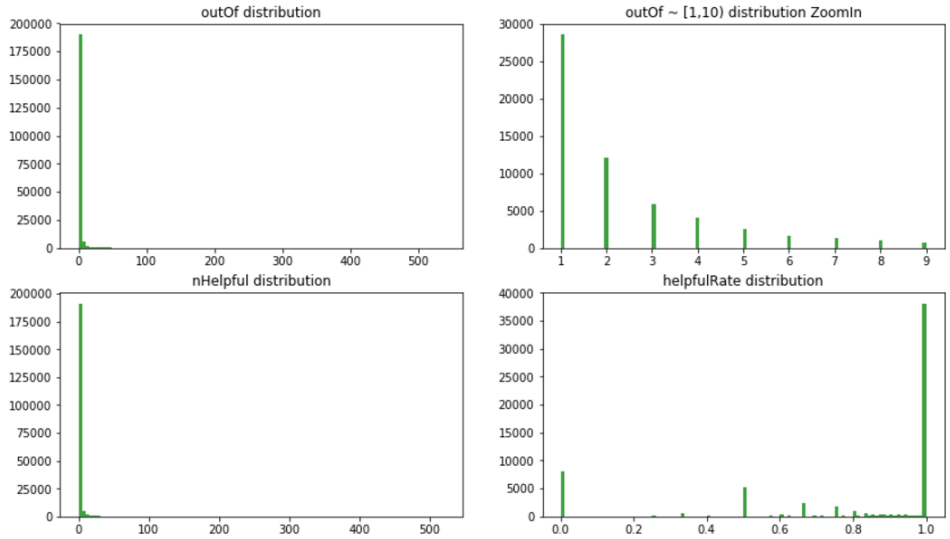
The code is submitted at github

https://github.com/mas-dse/w9yan/blob/master/DSE220/homeworks/Final/final.ipynb

1. Data exploration

Checking the distribution of target helpfulRate and various raw features, noticed 'out of' was very skewed with a long tail, and higher 'rating' turns more likely to get better helpfulRate. The 2 are most interesting features.

And 'price' has a lot missing value, categoryID/categories are highly biased on training data, 'summary' is not distinguishable between helpful and unhelpful. So these features were dropped.



2. Data processing:

2.1 Original features directly from data set

`rating, outOf`

`reviewText, unixReviewTime`: not used directly

2.2 Features developed

Here's a list of including developed features, ordered by the correlation to helpfulRate, as plotted in the notebook. Details of final used features are explained in next section.

```
['rating', 'itemRatingDev', 'itemRate', 'userRate', 'outOf', 'itemOutOfRat
io1', 'userAvgOutOf', 'itemAvgRating', 'itemOutOfRatio', 'itemAvgReviewLen
gthRatio', 'itemAvgOutOf', 'reviewLength', 'itemReviewLengthRatio', 'userR
eviewCount1', 'itemReviewCount1', 'userReviewCount', 'itemReviewTimePast',
 'itemReviewCount', 'itemReviewTimeRatio']
```

2.3 Cleaning Outliers

Filters with various features based on the scatter plot which remove the obvious outliers. Finally, I only used b elow filters as their features were used in final model.

`outOf > 0`: since helpfulRate is unknown for these samples, and we know nHelpful=0 for these samples.

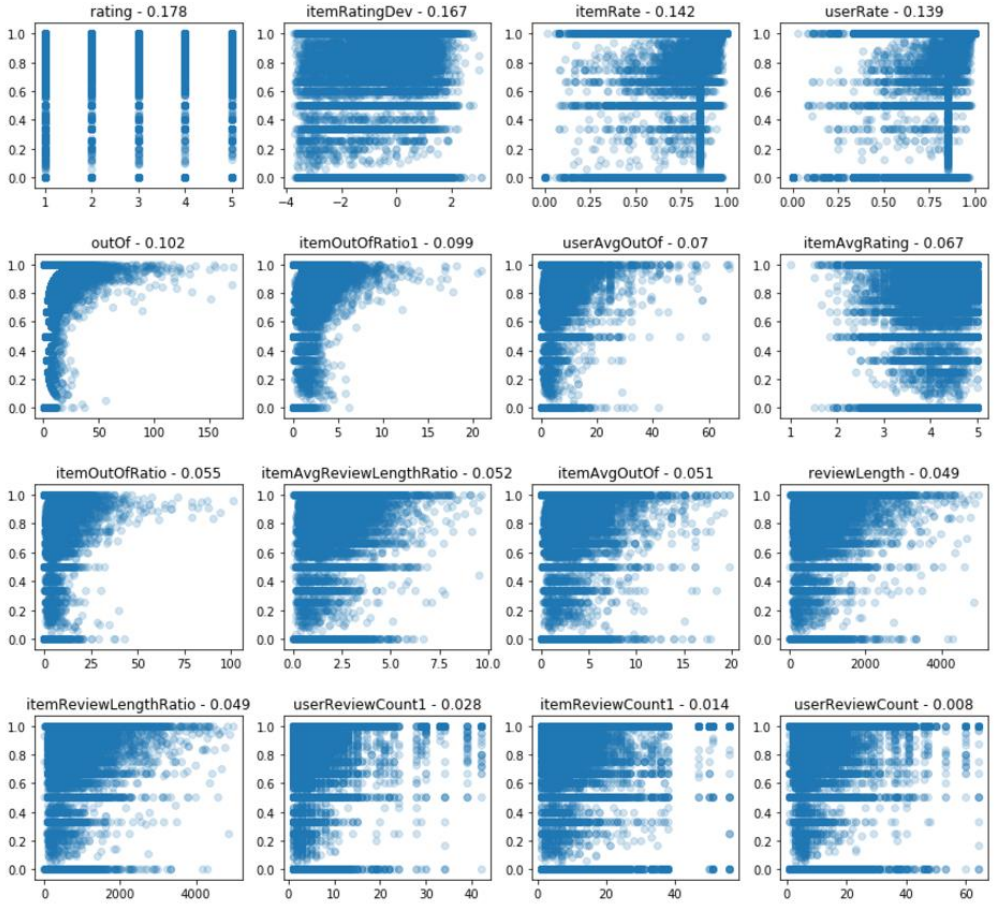
`outOf < 500`: remove those extremely popular reviews

`reviewLength < 7500`: remove reviews that are too long

3. Feature Selection:

Initial selection of features is based on correlation to helpfulRate, and the result with linear regression model on valid dataset is roughly aligned with the order of correlation. Latter features in the list really make little difference, and only 6 or 7 out of total were selected for model tuning.

Below scatter plotting shows correlation to helpfulRate from part of the features listed previously.



The features 'userRate', 'itemRate' show high correlation to helpfulRate but have problem on overfitting to training data. The correlation values are already improved version by some workaround, but they still make test result worse. They're dropped finally since I failed to work around the overfitting.

Below 7 features were used for model tuning later:

`rating, outOf, reviewLength,`

`itemRatingDev`: rating deviation from average rating of the item, signed integer.

`itemReviewTimePast`: unixReviewTime – time of first review of the item

`itemAvgReviewLengthRatio`: reviewLength over average reviewLength of the item

`itemOutOfRatio1`: item's outOf over the average outOf of the item's reviews which has outOf>0.

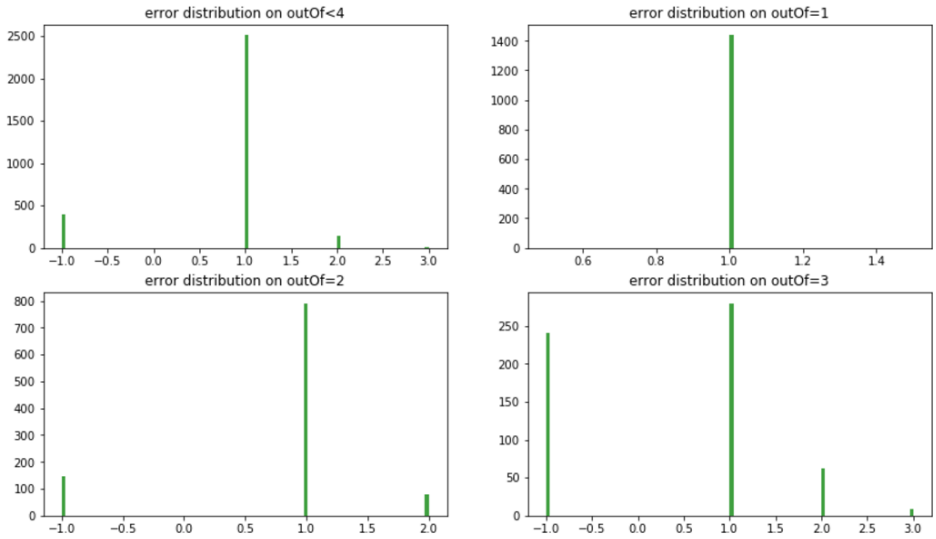
5 used by final model: `rating, reviewLength, itemRatingDev, outOf, itemReviewTimePast`

4. Modeling

4.1 One regression model

LinearRegression, Ridge, Lasso, SVR, GradientBoostingRegressor were ever tried, and not much difference in validation result. GradientBoostingRegressor is a bit better than others, while LinearRegressoin is mainly used for feature selection and segmentation experiment(in later section) since it's simpler and faster. LinearRegression model's parameters (fit_intercept=True, normalize=True) is used, so it can be directly fed with original feature values.

Then by plotting the distribution of errors as well as errors versus the key features, it shows one LinearRegression model performs bad on portion of small outOf, eg It predicted all 100% helpfulRate for outOf=1,2. The problem is related to the long-tailed the distribution of outOf.



4.2 Multiple regression models with multiple segments

First try is to separate dataset with outOf=1 to a classification problem, DecisionTree and SVM were used, they show a bit improvement on validation dataset but didn't get good result on testing. I gave up this approach finally.

Then, 2 segmentations were tried with cutting at 8,10,15,30 each time and 2 same LinearRegression models was used to fit each segment. Validation dataset was splitted on same threshold. And once it's proved to perform better than one single model, more segmentations were enumerated.

To find the best segmentation, 3 loops of combinations were involved to evaluate the performance. First loop enumerates different feature sets, 2nd loop enumerate different segment count and thresholds from 2 up to 5 segments, and 3rd loop tried different models including LinearRegression and GradientBoostingRegressor with different parameters.

Finally, the best performance I can achieve is combined model of 4 segments splitting at (2,10,30).

a. filter: only data in this range will be used for final model training.

```
outOf > 0
outOf < 500
reviewLength < 7500
```

b. features:

```
['rating', 'reviewLength', 'itemRatingDev', 'outOf', 'itemReviewTimePast']
```

c. segments and model for each segment

```
Segment1: outOf<=2
GradientBoostingRegressor(n_estimators=120, learning_rate=0.1,
            max_depth=3, random_state=0, loss='ls'
Segment2: outOf>2 & out of<=10
GradientBoostingRegressor(n_estimators=140, learning_rate=0.1,
            max_depth=5, random_state=0, loss='ls'
Segment3: outOf>10 & out of<=30
GradientBoostingRegressor(n_estimators=110, learning_rate=0.1,
            max_depth=3, random_state=0, loss='ls'
Segment4: outOf>30
GradientBoostingRegressor(n_estimators=110, learning_rate=0.1,
            max_depth=3, random_state=0, loss='ls'
```