Capstone Project

Pradeep Pujari

Machine Learning Engineer Nanodegree

June 3, 2016

Definition

Project Overview:

This project work is from yelp data challenge. Many businesses collect reviews of their services or products. Along with reviews text, reviewer is also asked to enter rating in the range of one to five, five being excellent. Sometimes, there is discrepancy between the numeric value of the rating and reviews text. In fact many start-ups evolved to understand and interpret this data and provide good insight to the business. Yelp provides data set for this challenge. In their data challenge page, they also put some ideas to consider. I have chosen the idea of predicting numeric rating from the reviews text alone. They also provided example code to use for this datasets.

Problem Statement

Predict numeric rating from reviews text. Sometimes, it is seen that there might be some anomaly between length of the reviews text and given numeric rating. Human intuition about expressing reviews text into number varies from person to person.

We are given labeled training data, which has reviews text, and it's rating. Although the rating is for that particular product or service, here I have not considered that relationship. Reviews text is tokenized with TF-IDF Vector. Then Logistic Regression Classifier is used to train the model. I persist the model with pickle. The predict API is used to predict for unseen text.

Metrics

Precision, recall and F1 score is used to measure the performance of the model. Model selection and evaluation using tools, such as **grid_search.GridSearchCV** and**cross_validation.cross_val_score**, take a scoring parameter that controls what metric they apply to the estimators evaluated. All scorer objects follow the convention that higher return values are better than lower return values. Thus the returns from mean_absolute_error and mean_squared_error, which measure the distance between the model and the data, are negated.

Since this is a logistic classifier, F1_macro which is a weighted average of precision and recall, is used.

Analysis

Data Exploration

Yelp business review data set consists five data files. For this project, I considered only reviews file. Other remaining data files are not required. Data is in JSON format with the following structure.

```
review
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

This data file has 2225213 reviews. I could not take all of them, as it gives out of memory error during evaluation and training. Reviews are for a business or any properties of business. Business types range from restaurants to home improvement to automotive services. I simply used TFIDF vector of the text. I do see new line character \n in most of the place and were removed. Trained the model with 50000 reviews data.

Data Format

```
business
{
    'type': 'business',
    'business_id': (encrypted business id),
    'name': (business name),
    'neighborhoods': [(hood names)],
    'full_address': (localized address),
    'city': (city),
    'state': (state),
    'latitude': latitude,
    'longitude': longitude,
    'stars': (star rating, rounded to half-stars),
    'review_count': review count,
    'categories': [(localized category names)]
    'open': True / False (corresponds to closed, not business hours),
    'hours': {
        (day_of_week): {
            'open': (HH:MM),
            'close': (HH:MM)
        },
        ...
    },
    'attributes': {
        (attribute_name): (attribute_value),
        ...
    },
}


review
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}



user
{
    'type': 'user',
    'user_id': (encrypted user id),
    'name': (first name),
    'review_count': (review count),
    'average_stars': (floating point average, like 4.31),
    'votes': {(vote type): (count)},
    'friends': [(friend user_ids)],
    'elite': [(years_elite)],
    'yelping_since': (date, formatted like '2012-03'),
    'compliments': {
        (compliment_type): (num_compliments_of_this_type),
        ...
    },
    'fans': (num_fans),
}
check-in
{
    'type': 'checkin',
    'business_id': (encrypted business id),
    'checkin_info': {
        '0-0': (number of checkins from 00:00 to 01:00 on all Sundays),
        '1-0': (number of checkins from 01:00 to 02:00 on all Sundays),
        ...
        '14-4': (number of checkins from 14:00 to 15:00 on all Thursdays),
        ...
        '23-6': (number of checkins from 23:00 to 00:00 on all Saturdays)
    }, # if there was no checkin for a hour-day block it will not be in the dict
}
tip
{
    'type': 'tip',
    'text': (tip text),
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'date': (date, formatted like '2012-03-14'),
    'likes': (count),
}
```

Exploratory Visualization

I have attached explore_data ipython notebook.

```
Total documents in the data set: 2225213

1 rated reviews = 11.71%
2 rated reviews = 8.54%
3 rated reviews = 12.68%
4 rated reviews = 26.59%
5 rated reviews = 40.49%
```

Algorithms and Techniques

We have an array of algorithms like Naive Bayes, decision trees, Random Forests, Support Vector Machines, and many others. Logistic regression classifier classifies text documents by using a bag-of-words approach, efficiently handles sparse features. I have chosen Logistic Regression, as it is fast, easy to regularize and it outputs well-calibrated predicted probabilities. It also works with categorical features.

Raw reviews data cannot be directly feed to Logistic Regression algorithm as it expects numerical feature vector with a fixed size rather than the raw text documents of variable length. In order to address this, sci-kit learn provides utilities to extract numerical features from text content, namely

- **tokenizing** strings and giving an integer id for each possible token, for instance by using white-spaces and punctuation as token separators.

- **counting** the occurrences of tokens in each document.
- **normalizing** and weighting with diminishing importance tokens that occur in the majority of samples / documents.

I used tf-idf vectorizer to generate features from reviews text. I set the parameter to remove English stop words.

Tuned Parameters

min_df : float in range [0.0, 1.0] or int, default=1 (Tuned to 0.1)

When building the vocabulary ignore terms that have a document frequency strictly lower than the given threshold. This value is also called cut-off in the literature. If float, the parameter represents a proportion of documents, integer absolute counts. This parameter is ignored if vocabulary is not None.

Stop word English is used.

Simple Logistic Regression Classifier with GridSearchCV to train the model. GridsearchCV searches over parameter values for logistic regression. Parameters that are not directly learnt, also called hyper parameters can be set by searching a parameter space for the best score. A search consists of:

- a classifier
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme; and
- a score function.

By default, parameter search uses the score function of the estimator to evaluate a parameter setting.

Following parameters were tuned with GridsearchCV

```
grid = {'C': 10.0 ** np.arange(-2, 3),
                        'penalty': ['l1', 'l2'],
                        'class_weight': [None, 'auto']}
```

C Inverse of regularization strength

Penalty whether L1 or L2 regularization

class_weight	Weights associated with classes

Methodology

Data Preprocessing

During preprocessing, I extracted only relevant fields required for the algorithm. They are

Review_id

Review_text

Votes (funny, useful and cool)

Stars (i.e. numeric rating) this is the class label.

Each observation in this dataset is a review of a particular business by a user.

Added a new feature, reviews text length. Instead of absolute length, I made it a range, because adding absolute length in fact reduced precision and recall.

Explored the relationship between each of the vote types (cool/useful/funny) and the number of stars. "stars" field is a categorical variable. Looking the difference between each vote types.

Implementation

Used logistic regression classifier. Performance matrics used are

Accuracy

F1-Matric (Precision, Recall and F1 Score)

Confusion Matrix

Refinement

Actually, after adding text length feature, model performance did not increase.

Results

Model Evaluation and Validation

Evaluated the model by splitting it into training and testing sets 80:20 and K-fold validation

```
cv = KFold(self.x_train.shape[0], n_folds=5, shuffle=True, random_state=0)
```

random_state: When shuffle=True, pseudo-random number generator state used for shuffling. If None, use default numpy RNG for shuffling.

shuffle : Whether to shuffle the data before splitting into batches.

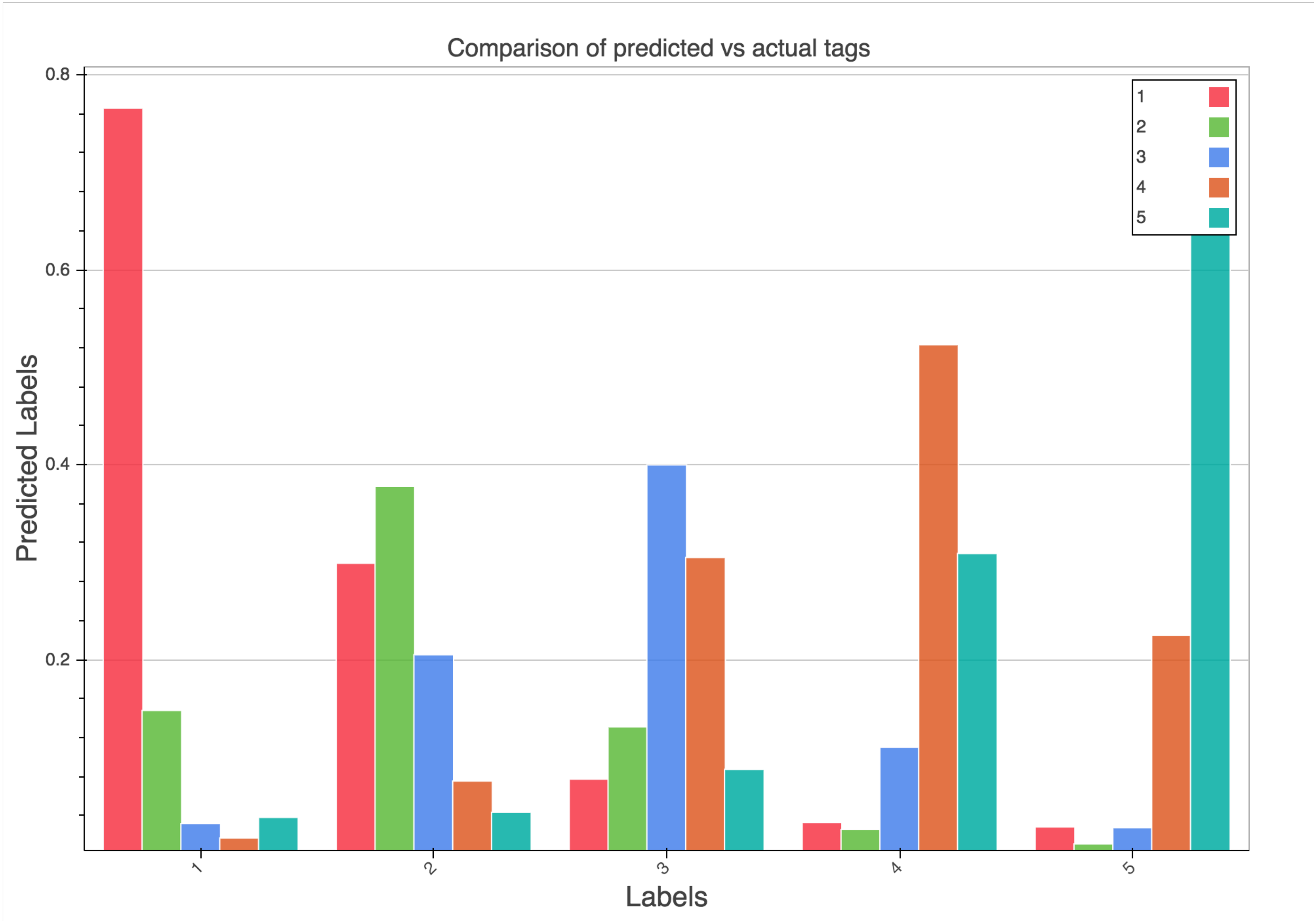
```
2016-06-09 12:29:23.598000 INFO extractors.supervised_extractors.train.Trainer.train:306
[MainThread] Best estimator LogisticRegression(C=1.0, class_weight='auto', dual=False,
fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr',
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0)
2016-06-09 12:29:23.706000 INFO extractors.supervised_extractors.train.Trainer.evaluate:322
[MainThread] Precision on training data = 0.82665625
2016-06-09 12:29:23.710000 INFO extractors.supervised_extractors.train.Trainer.evaluate:323
[MainThread] Precision on test data = 0.577
2016-06-09 12:29:23.715000 INFO extractors.supervised_extractors.train.Trainer.evaluate:325
[MainThread] Accuracy on training data = 0.82665625
2016-06-09 12:29:23.716000 INFO extractors.supervised_extractors.train.Trainer.evaluate:326
[MainThread] Accuracy on test data = 0.577
2016-06-09 12:29:23.720000 INFO extractors.supervised_extractors.train.Trainer.evaluate:328
[MainThread]
 Classification Report
              precision    recall  f1-score   support

          1       0.60      0.77      0.67       948
          2       0.43      0.38      0.40       810
          3       0.50      0.40      0.44      1306
          4       0.54      0.52      0.53      2348
          5       0.67      0.71      0.69      2588
```

```
avg / total       0.57      0.58      0.57      8000

 [MainThread] Printing model stats
2016-06-09 12:29:23.766000 INFO extractors.supervised_extractors.train.Trainer.__visualize:375
[MainThread] Overall precision = 0.577
2016-06-09 12:29:23.766000 INFO extractors.supervised_extractors.train.Trainer.__visualize:379
[MainThread] At threshold 0.5    Precision = 0.762895  Recall = 0.24775
2016-06-09 12:29:23.767000 INFO extractors.supervised_extractors.train.Trainer.__visualize:379
[MainThread] At threshold 0.7    Precision = 0.940789  Recall = 0.053625
2016-06-09 12:29:23.767000 INFO extractors.supervised_extractors.train.Trainer.__visualize:379
[MainThread] At threshold 0.9    Precision = 1.0  Recall = 0.00025
2016-06-09 12:29:24.334000 INFO extractors.supervised_extractors.train.Trainer.__visualize:391
[MainThread]

confusion matrix
[[ 726  242  101   77   73]
 [ 140  306  171   60   28]
 [  30  166  522  258   71]
 [  16   61  398 1228  582]
 [  36   35  114  725 1834]]
Done
```

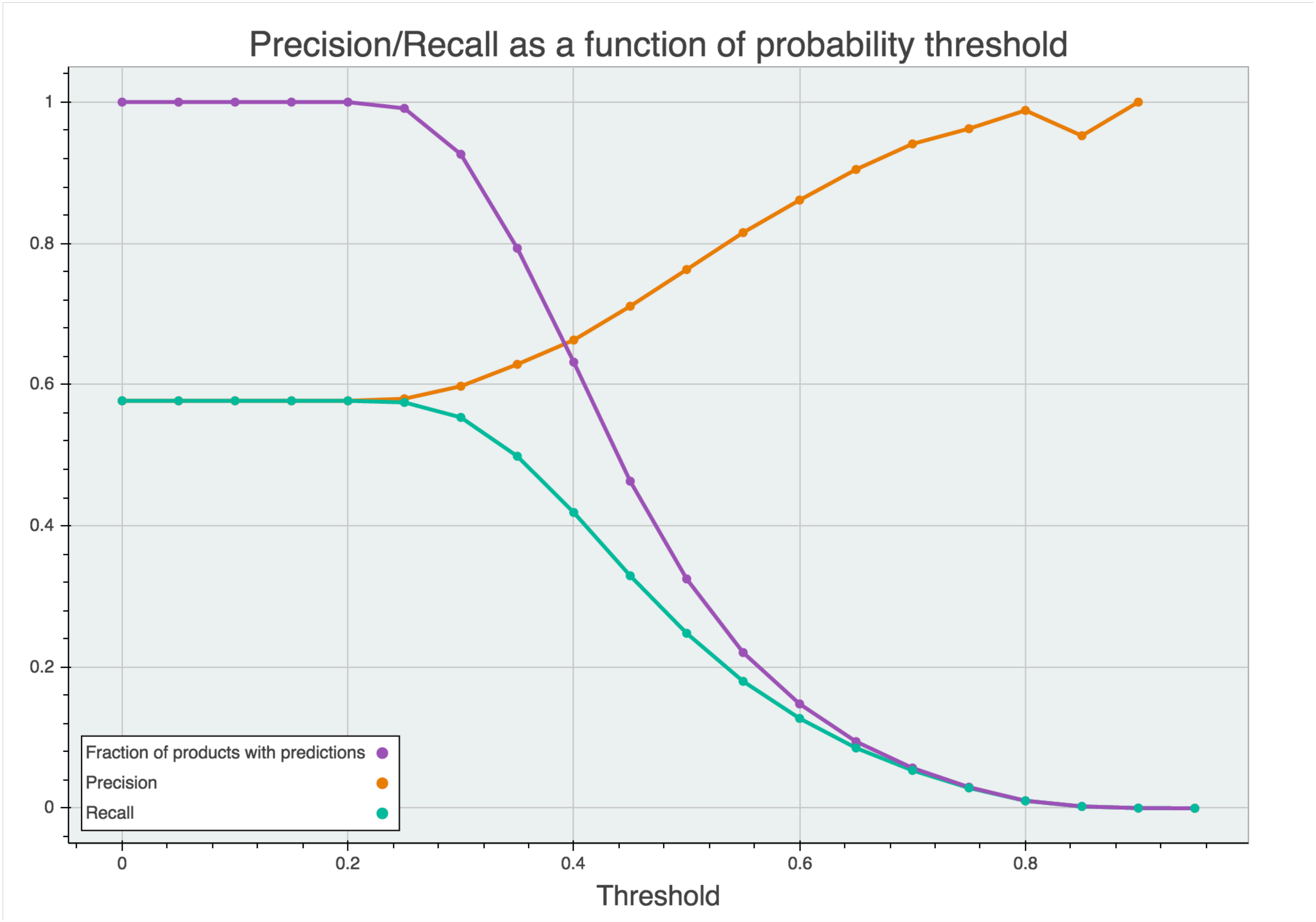
Justification

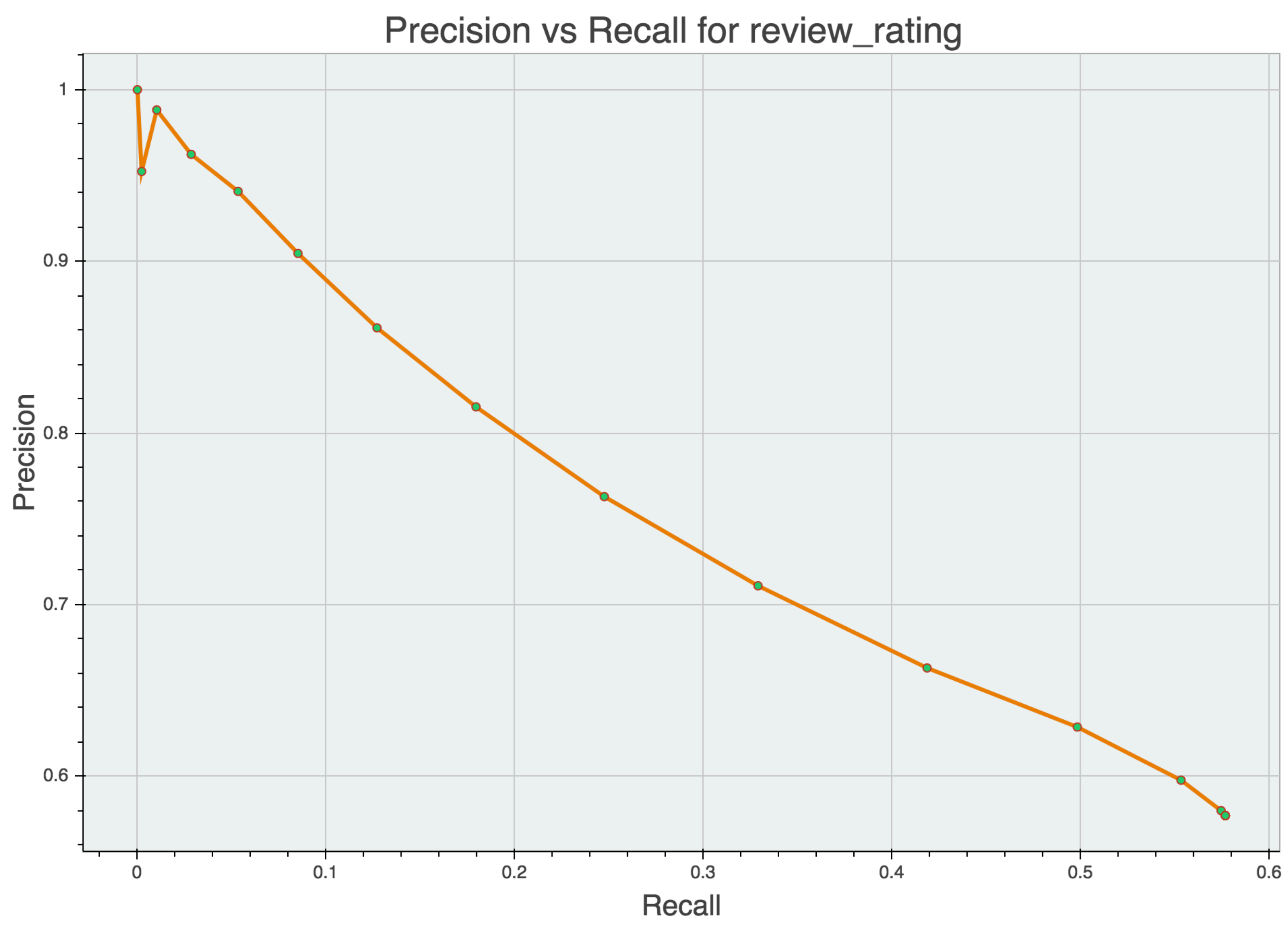
Conclusion

Free-Form Visualization



Comparison of predicted vs actual tags
Predicted Labels
0.8
0.6
0.4
0.2
Labels
1
2
3
4
5
1
2
3
4
5


Precision/Recall as a function of probability threshold
Fraction of products with predictions
Precision
Recall
Threshold
0
0.2
0.4
0.6
0.8
1


Precision vs Recall for review_rating
Precision
Recall
1
0.9
0.8
0.7
0.6
0
0.1
0.2
0.3
0.4
0.5
0.6

Reflection

In conclusion, adding new features did not improve accuracy. It might work to find out popular words in one rating reviews and five review ratings and use them as feature, might improve model performance.

Reference:

Yelp Data challenge

https://www.yelp.com/dataset_challenge

Get start code
https://github.com/Yelp/dataset-examples