Project Report for Predicting Ratings from Text

COMP8745 Course Project

Laqin Fan

1 Project Introduction

In this project, the purpose is to predict rating scores(1-5) from the text in online reviews. Given the dataset with reviews and their rating scores, I will take the review text as an input, and predict what is the rating score for that review. Meanwhile, I experiment with several supervised learning algorithms using this dataset.

2 Dataset

There are 2 files, each contains review-text and the score separated by a tab, these reviews are taken from outlier detection dataset: http://odds.cs.stonybrook.edu.

1. traing.txt, 10k reviews
2. test.txt, 1k reviews

3 Surpervised Learners

Use the following learners to train the review text and predict the rating score:

1. Neural networks (MLPClassifier in sklearn)
2. Nave Bayes (MultinomialNB in sklearn)

3. Logistic Regression (LogisticRegression in sklearn)

4. AdaBoosting (AdaBoostClassifier in sklearn)

5. SVM (svm.svc in sklearn)

4 Results for Each Task

4.1 Task 1 Result

Run 5-fold Cross Validation on the training.txt using above 5 learning algorithms. Report the average-precision, average-recall and average-F1-scores.
1. Neural networks: In neural networks change the number of hidden layers and number of units in each layer, the optimal setting: hidden size (70, 50, 100)

```
(70, 50, 100)  average cv score: 0.5269095480941693
(70, 50, 100)  average precision score: 0.5399264951553625
(70, 50, 100)  average recall score: 0.5399264951553625
(70, 50, 100)  average f1 score: 0.5399264951553625
```

2. Nave Bayes:

```
Naive Bayes Classifier:
average cv score: 0.42736842105263156
average precision score: 0.506
average recall score: 0.506
average f1 score: 0.506
```

3. Logistic Regression:In Logistic regression change the penalty: L1 regularization that can also perform feature selection and L2. Also change the regularization strength parameter(C), the optimal setting: ('l2', 1.0)

```
('l1', 2.0)  average cv score: 0.4606516290726817
('l1', 2.0)  average precision score: 0.509
('l1', 2.0)  average recall score: 0.509
('l1', 2.0)  average f1 score: 0.509
('l1', 3.0)  average cv score: 0.4542355889724311
('l1', 3.0)  average precision score: 0.49
('l1', 3.0)  average recall score: 0.49
('l1', 3.0)  average f1 score: 0.49
```

```
('l2', 1.0)  average cv score: 0.4670676691729323
('l2', 1.0)  average precision score: 0.535
('l2', 1.0)  average recall score: 0.535
('l2', 1.0)  average f1 score: 0.535
('l2', 2.0)  average cv score: 0.46335839598997497
('l2', 2.0)  average precision score: 0.527
('l2', 2.0)  average recall score: 0.527
('l2', 2.0)  average f1 score: 0.527
('l2', 3.0)  average cv score: 0.4608521303258145
('l2', 3.0)  average precision score: 0.521
('l2', 3.0)  average recall score: 0.521
('l2', 3.0)  average f1 score: 0.521
```

4. AdaBoosting: In Adaboosting change the number of estimators, the optimal setting: number of estimator is 50

```
n_estimators:  10  average cv score: 0.39458646616541354
n_estimators:  10  average precision score: 0.41
n_estimators:  10  average recall score: 0.41
n_estimators:  10  average f1 score: 0.41
n_estimators:  50  average cv score: 0.4140350877192982
n_estimators:  50  average precision score: 0.47
n_estimators:  50  average recall score: 0.47
n_estimators:  50  average f1 score: 0.47
n_estimators:  100  average cv score: 0.4109273182957393
n_estimators:  100  average precision score: 0.443
n_estimators:  100  average recall score: 0.443
n_estimators:  100  average f1 score: 0.443
```

5. SVM: In SVMs, change the penalty parameter C and the kernel type, the optimal setting: (1.0, 'linear')

```
(1.0, 'linear')  average cv score: 0.4719799498746868
(1.0, 'linear')  average precision score: 0.528
(1.0, 'linear')  average recall score: 0.528
(1.0, 'linear')  average f1 score: 0.528
(1.0, 'poly')  average cv score: 0.2693734335839599
(1.0, 'poly')  average precision score: 0.301
(1.0, 'poly')  average recall score: 0.301
(1.0, 'poly')  average f1 score: 0.301
```

```
(2.0, 'linear')  average cv score: 0.46466165413533833
(2.0, 'linear')  average precision score: 0.515
(2.0, 'linear')  average recall score: 0.515
(2.0, 'linear')  average f1 score: 0.515
(2.0, 'poly')  average cv score: 0.2687719298245614
(2.0, 'poly')  average precision score: 0.306
(2.0, 'poly')  average recall score: 0.306
(2.0, 'poly')  average f1 score: 0.306
(3.0, 'linear')  average cv score: 0.45513784461152884
(3.0, 'linear')  average precision score: 0.517
(3.0, 'linear')  average recall score: 0.517
(3.0, 'linear')  average f1 score: 0.517
```

4.2 Task 2 Result

Perform feature selection using PCA and re-run the algorithms with their optimal settings. But the text-term matrix has a lot of sparse input, and PCA does not support sparse input, therefore here I am using TruncatedSVD instead to reduce the dimentionality.NN: neural networks, LR: logistic Regression, AB: Adaboosting, SVC: SVM. In this case, we can see that, the n-component is 10, the accuracy is the highest.

```
NN, n_component:  2  average cv score: 0.23959899749373434
LR, n_component:  2  average cv score: 0.22175438596491226
AB, n_component:  2  average cv score: 0.2368922305764411
SVC, n_component:  2  average cv score: 0.22716791979949874
NN, n_component:  3  average cv score: 0.3267167919799499
LR, n_component:  3  average cv score: 0.27729323308270676
AB, n_component:  3  average cv score: 0.29874686716791976
SVC, n_component:  3  average cv score: 0.2747869674185464
NN, n_component:  4  average cv score: 0.36300751879699245
LR, n_component:  4  average cv score: 0.3415538847117795
AB, n_component:  4  average cv score: 0.3387468671679198
SVC, n_component:  4  average cv score: 0.34155388471177944
NN, n_component:  5  average cv score: 0.3651127819548872
LR, n_component:  5  average cv score: 0.3499749373433584
AB, n_component:  5  average cv score: 0.33974937343358397
SVC, n_component:  5  average cv score: 0.3538847117794486
```

```
NN, n_component:  10  average cv score: 0.4150375939849624
LR, n_component:  10  average cv score: 0.39859649122807017
AB, n_component:  10  average cv score: 0.3757393483709273
SVC, n_component:  10  average cv score: 0.40812030075187966
```

4.3 Task 3 Result

Use the sentiment words in pos-words.txt and neg-words.txt to filter words from the review text, and then evaluate the algorithms once again.

Result: the accuracy doesn't become better

```
[NN, NB, LR, AB, SVC]
average cv score: 0.3390121411120486
average precision score: 0.35
average recall score: 0.35
average f1 score: 0.35
average cv score: 0.3972283190137029
average precision score: 0.4066666666666667
average recall score: 0.4066666666666667
average f1 score: 0.4066666666666667
average cv score: 0.408597390891563
average precision score: 0.38333333333333336
average recall score: 0.38333333333333336
average f1 score: 0.38333333333333336
average cv score: 0.30872142042354805
average precision score: 0.2966666666666667
average recall score: 0.2966666666666667
average f1 score: 0.2966666666666667
average cv score: 0.4142724404796561
average precision score: 0.3566666666666667
average recall score: 0.3566666666666667
average f1 score: 0.35666666666666674
```

4.4 Task 4 Result

Perform evaluation on the test dataset using the optimal parameter settings that were obtained from the training set.

```
[NN, NB, LR, AB, SVC]
average cv score: 0.4927638277906814
average precision score: 0.4666666666666667
average recall score: 0.4666666666666667
average f1 score: 0.4666666666666667
average cv score: 0.502825343795269
average precision score: 0.5
average recall score: 0.5
average f1 score: 0.5
average cv score: 0.502928277174745
average precision score: 0.5133333333333333
average recall score: 0.5133333333333333
average f1 score: 0.5133333333333333
average cv score: 0.3487336310415127
average precision score: 0.37666666666666665
average recall score: 0.37666666666666665
average f1 score: 0.37666666666666665
average cv score: 0.5085199555434283
average precision score: 0.49666666666666665
average recall score: 0.49666666666666665
average f1 score: 0.49666666666666665
```

For the best performing algorithm, compute precision and recall for every rating score separately. From the result, the rating score of the review is 3, the precision and recall is not good. That means if the the reviews are neural, don't contain sentiment(positive or negative) words, then they are hard to predict the rating score.

```
precision score:
[0.57142857 0.51612903 0.27777778 0.37096774 0.71186441]
recall score:
[0.52941176 0.47058824 0.30612245 0.60526316 0.54545455]
```

4.5 Task 5 Result

Discuss some ideas that could help improve your prediction.

1. Use sentiment analysis, we capture semantic similarities between words,

and capture the sentiments of individual words within a review, in this way, it may help improve the accuracy of prediction.

2. Use natural language processing method, such as Latent Semantic Analysis, LSA can finds hidden relationship between words in order to improve information understanding.