Adithyan Subramanian

Machine Learning report

Mentor: Mukesh Mithrakumar

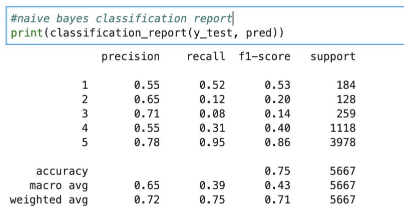
The final goal of this project was to predict Amazon Review Ratings based on the text from the reviews. In the previous milestone report, we explored the cleaning and prep work to get the dataset prepared for machine learning. Now, we will discuss the machine learning techniques used in this project.

Processing

To start we split the dataset into train and test sets, with an 80/20 split. When we had the train and test sets, we then initialized a countervectorizer and tfidfvectorizer to be used for machine learning. We used the vectorizers on the train and test set and created dataframes to be used for machine learning.

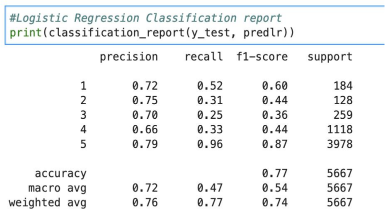
Initial modeling

In order to determine which model would be best for this dataset, we initially created three models: a Naive Bayes model, a logistic regression model, and a decision tree model. The classification report for all three models can be seen below



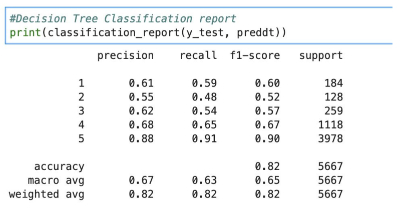
```
#naive bayes classification report
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           1       0.55      0.52      0.53       184
           2       0.65      0.12      0.20       128
           3       0.71      0.08      0.14       259
           4       0.55      0.31      0.40      1118
           5       0.78      0.95      0.86      3978

    accuracy                           0.75      5667
   macro avg       0.65      0.39      0.43      5667
weighted avg       0.72      0.75      0.71      5667
```



```
#Logistic Regression Classification report
print(classification_report(y_test, predlr))
```

```
              precision    recall  f1-score   support

           1       0.72      0.52      0.60       184
           2       0.75      0.31      0.44       128
           3       0.70      0.25      0.36       259
           4       0.66      0.33      0.44      1118
           5       0.79      0.96      0.87      3978

    accuracy                           0.77      5667
   macro avg       0.72      0.47      0.54      5667
weighted avg       0.76      0.77      0.74      5667
```



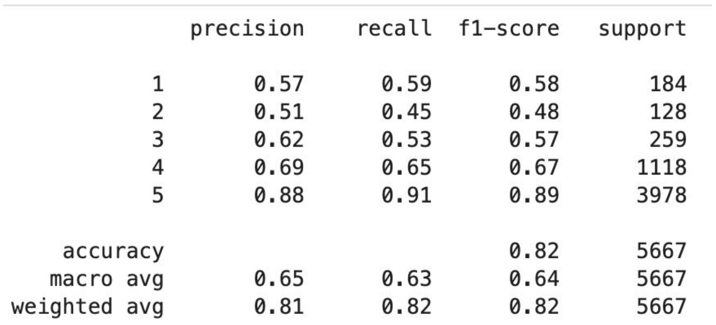
```
#Decision Tree Classification report
print(classification_report(y_test, preddt))
```

```
              precision    recall  f1-score   support

           1       0.61      0.59      0.60       184
           2       0.55      0.48      0.52       128
           3       0.62      0.54      0.57       259
           4       0.68      0.65      0.67      1118
           5       0.88      0.91      0.90      3978

    accuracy                           0.82      5667
   macro avg       0.67      0.63      0.65      5667
weighted avg       0.82      0.82      0.82      5667
```

As can be seen, the decision tree model had the most accuracy, so we chose to focus our efforts on optimizing that model.

Model optimization

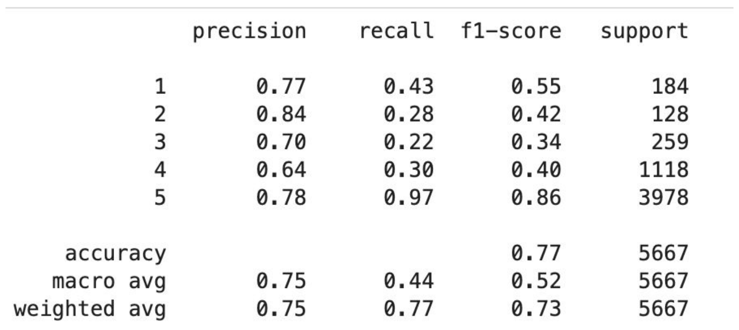
Using a GridSearchCV to find the best parameters, we found that the most accurate model, would be one with a max_depth of 5000, and a minimum samples split of 2. This model gave us a training set accuracy of 0.984, and a test set accuracy of 0.818. The classification report can be seen to the left



```
              precision    recall  f1-score   support

           1       0.57      0.59      0.58       184
           2       0.51      0.45      0.48       128
           3       0.62      0.53      0.57       259
           4       0.69      0.65      0.67      1118
           5       0.88      0.91      0.89      3978

    accuracy                           0.82      5667
   macro avg       0.65      0.63      0.64      5667
weighted avg       0.81      0.82      0.82      5667
```

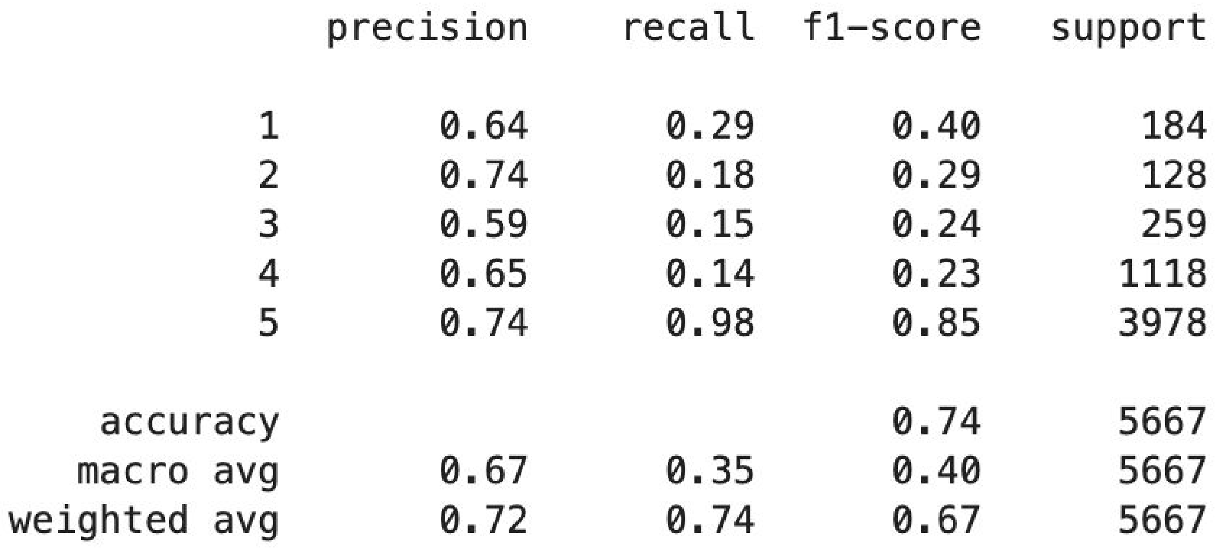
Although noting a high level of accuracy, the difference between the train test accuracy and test set accuracy was concerning for overfitting. We then decided

to alter the parameters to decrease the difference between training set accuracy and the test set accuracy in order to lower overfitting.
Initially, we attempted to introduce some noise to the model using a ridge classifier. This gave us a training set accuracy of 0.839 and a test set accuracy of 0.766. The classification report can be seen below.



```
              precision    recall  f1-score   support

           1       0.77      0.43      0.55       184
           2       0.84      0.28      0.42       128
           3       0.70      0.22      0.34       259
           4       0.64      0.30      0.40      1118
           5       0.78      0.97      0.86      3978

    accuracy                           0.77      5667
   macro avg       0.75      0.44      0.52      5667
weighted avg       0.75      0.77      0.73      5667
```

The difference between the training set accuracy and the testing set accuracy is still indicating overfitting, so we will return to the parameters of the decision tree, and see if we can decrease the difference.
After much trial and error, we were able to determine that the parameters with the least likeliness for overfitting were a maximum depth of 20, and a minimal samples split of two. This yielded a training set accuracy of 0.775 and a testing set accuracy of 0.736. The classification report can be seen below.

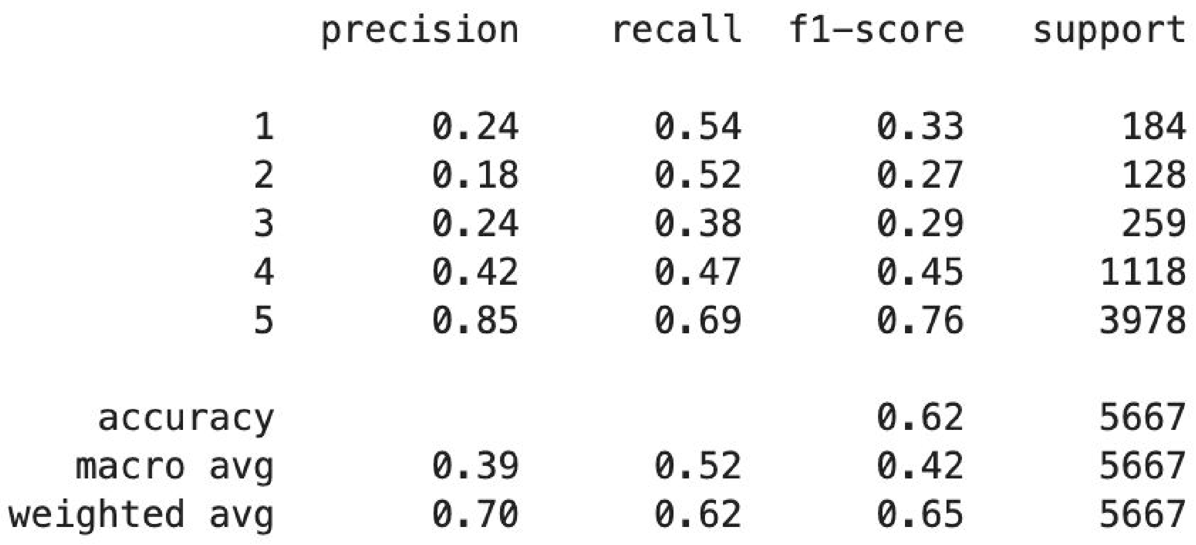


```
              precision    recall  f1-score   support

           1       0.64      0.29      0.40       184
           2       0.74      0.18      0.29       128
           3       0.59      0.15      0.24       259
           4       0.65      0.14      0.23      1118
           5       0.74      0.98      0.85      3978

    accuracy                           0.74      5667
   macro avg       0.67      0.35      0.40      5667
weighted avg       0.72      0.74      0.67      5667
```

This seems to have solved the overfitting issue, but now the accuracy is low. As mentioned in the previous status report, the dataset is heavily skewed towards 5-star reviews. In order to compensate for this, we decided to employ some oversampling methods.

Oversampling

We used the SMOTE method to equalize all review ratings from 1 to 5 stars. To accomplish this we used the ‘not majority’ setting to resample all reviews with 1-4 star ratings. Pairing this resampling with the ridge classifier, we got a training set accuracy of 0.627 and a testing set accuracy of 0.624. The classification report can be seen below.



```
              precision    recall  f1-score   support

           1       0.24      0.54      0.33       184
           2       0.18      0.52      0.27       128
           3       0.24      0.38      0.29       259
           4       0.42      0.47      0.45      1118
           5       0.85      0.69      0.76      3978

    accuracy                           0.62      5667
   macro avg       0.39      0.52      0.42      5667
weighted avg       0.70      0.62      0.65      5667
```

This completely solved the overfitting issue, but now the accuracy was very low. This is not an ideal model, so we chose to recommend the decision tree with a maximum depth of 20, and a minimal samples split of two as the best model to predict ratings based on review text.

Next Steps

Some next steps for this project would be to get more data on low rated reviews. The lack of 1 and 2-star reviews hamstrung any model that was attempted. It may help to scrape for 1 and 2-star reviews on Amazon in order to get a more diverse dataset. Finally, more time could be spent on both feature and parameter optimization.

Conclusion

The goal of this project was to predict amazon review ratings based on the text of the review. Our chosen model had an accuracy of 0.74. This is significantly better than an expected pure chance accuracy of 0.20. Due to the uneven data set, 5-star reviews had a very high accuracy of 0.85. The rest of the rating classifications were also above chance. We consider this evidence of a successful model. The next steps would be to introduce a more diverse dataset and to further optimize the model.