I began by parsing the review text and star ratings out of the all.review file. I then converted the star rating to a pos/neg label and preprocessed the text by removing punctuation, numbers, and converting to lowercase. Additionally I removed stop-words and words that appear less than 3 times in the vocabulary. I then vectorized the reviews using one-hot embeddings over the vocabulary.

Here is an example of a review before and after cleaning them up:

Before: *I purchased 3 different Embassy purses last December. None of them were the quality I had hoped for, and none were used as the gifts they were intended for. In this case, the price should be a clue that the bags are cheaply made and not nearly as nice as they look in the pictures used. I would not buy from this company again*

After: *i purchased different embassy purses last december none of them were the quality i had hoped for and none were used as the gifts they were intended for in this case the price should be a clue that the bags are cheaply made and not nearly as nice as they look in the pictures used i would not buy from this company again*

For model selection I started with a linear regressor and an svc. The svc initially was performing better so I chose to run with that. I manually tuned its parameters, focusing on C and epsilon, until I believed my results were not going to improve much more. While tuning I used sklearn's cross-validation function to compare my results.

I was able to achieve an accuracy score of 90.05763688760807% on held-out test data.

When I tested my classifier on a different category of review (trained on 'apparel,' tested on 'baby'), the accuracy score reduced to 0.8517387218045113%. This is still a decent accuracy score which makes sense because the features that strongly indicate positivity or negativity (words like 'happy' or 'bad') are still going to work properly on different reviews. However, there could be some specific terminology used in one category that applies differently in a different context. Additionally, the vocabularies of the two different review sets are not exactly the same so the pretrained classifier may miss some words that do not appear in the vocabulary of the data it was trained on.

For better cross-category performance I could perhaps train a more involved model such as a transformer or neural net using word embeddings created with word2vec because with more layers of learning it could gain more transferrable knowledge than my more straightforward model.