In order to prepare for building a movie review classifier using scikit-learn, I first processed the data by cleaning it and removing punctuation. I then split the pre-sorted negative and positive reviews into training (70%), development (15%), and test (15%) sets. Once the data was cleaned and split up, I concatenated the negative and positive training reviews and created a list of their target values. I then wrote a function called set_features that would take in the training and development data and create a list of features (all words used in the reviews), removing words that occur less than n times and more than m times. I started with the values n = 5 and m = 5000 for these hyperparameters and tuned them on the development set later.

I then wrote a function called count_vectorizer that takes in the training data and vectorizes it, producing a numpy array for the count of each of the features in each training example. I passed the training data into this vectorizer and then used it to train a logistic regression model using scikit-learn. I then evaluated this model on the development set and it had an initial accuracy of about 74.3%. I adjusted the values of n and m in the set_features function to determine which values provided the highest accuracy. I found that n = 14 increased the accuracy to 75.3% and increasing or decreasing the value of m from 5000 did not improve the accuracy at all. Finally, I used both the training and development sets to train the model again and evaluated this model on the test set by first vectorizing the data using my count_vectorizer. The final accuracy of this model ended up being about 75.4%.

Next, I used scikit-learn's CountVectorizer to train a new logistic regression model on the data. I used the same process to tune the hyperparameters using the development set and found that any values of m (max_df) greater than 4500 provided the same accuracy of 75.5% on the development set, which was higher than any values lower than m = 4500. Changing the value from n = 5 (or min_df = 5) did not improve the accuracy so I kept this value the same. Finally, I

trained the model again using the concatenation of the training and development sets and evaluated the performance of this model on the test set. The classifier was about 74.1% accurate on the test set and this was very close to the accuracy of the model using my own count_vectorizer, although that model was slightly more accurate with an accuracy of about 75.4%. It is strange that the model where I used my own vectorizer was more accurate than scikit-learn's, so I wonder why this was the case.