Large Movie Review Dataset Sentiment Analysis Using Multinomial Naive Bayes Model:

A. Method:

Step 1: Data Input

The data is read from the training and testing folders separately into two respective pandas data frames. The columns of these data frames are the review text, and their corresponding label (1 if positive, 0 if negative). Both the data frames are then shuffled.

Step 2: Preprocessing

The review texts are cleaned by doing the following:

1. Lowercase the text
2. Remove symbols, digits, underscores, etc using regular expressions
3. Remove Stop-words using NLTK stop words
4. Lemmatisation to group together various forms of the same word using NLTK's WordNetLemmatizer.

Step 3: Feature extraction from the training data using the bags of word approach.

We convert each of our reviews to a vector containing frequency of each word (bag of word), using sklearn's CountVectorizer. Here, CountVectorizer converts each of our review text into a vector. Each column in that vector represents a word. The value in each of these column is the frequency of word (which the column represents) in the given review text.

We limit the number of words (features) that we use to form our extractor to the most common 10,000 words in our training dataset. This is done because if we include all the words in our data, we get a vector for each review with mostly null values. Different number of features were experimented, and using only the most common 10,000 words provided the best results.

Step 4: Model Training

Multinomial Naive Bayes Classifier from sklearn (MultinomialNB) is used for our sentiment analysis model. We use this classifier because it is a modified Naive Bayes Model which is better suited for discrete values like word counts, and our feature vectors contain discrete values of word counts. We first fit the model on our training data vectors, and get an accuracy of 86.7% on our training data.

Step 5: Model Testing

We first vectorise our testing data using the CountVectoriser, and then use our trained model to make predictions on the new testing data.

Step 6: Model Prediction

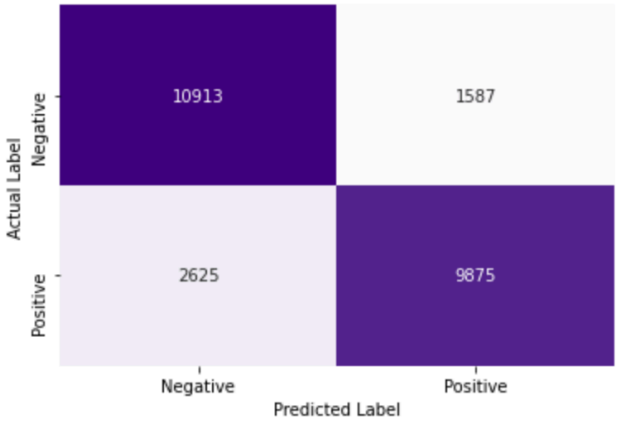
We then compare our predicted values with the actual labels, and observe that we get an accuracy of 83.1% on predicting the testing data.

B. Results:

Classification Report:

	precision	recall	f1-score	support
0	0.81	0.87	0.84	12500
1	0.86	0.79	0.82	12500
accuracy			0.83	25000
macro avg	0.83	0.83	0.83	25000
weighted avg	0.83	0.83	0.83	25000

Confusion Matrix:



True Positive: 9875/25000 = 39.5%
True Negative: 10913/25000 = 43.6%
False Positive: 1587/25000 = 6.3%
False Negative: 2625/25000 = 10.5%
Accuracy = 83.1% (39.5% + 43.6%)

We can observe that although we are getting an accuracy of 83.1% in predicting the actual label, we are still 16.9% times wrong in our prediction. We can observe that our precision is higher for prediction of positive reviews, but our recall is higher for prediction of negative reviews. Moreover, we can observe that the probability of occurrence of a false negative is higher than that of the occurrence of a false positive. This means that at times when our model predicts incorrectly, it predicts a positive review as negative more than the times when it predicts a negative review as positive. This could be due to some words which are more common in negative reviews are sometimes used by some users in positive reviews too (more often than vice versa).

The overall accuracy of the model is observed to be 83.1% on the testing data. This could be due to the following reasons:

1. Our testing data consists words which are not in our training data.
2. Since we use only the top 10,000 most frequent features while vectorising our data, it is possible that the most frequent features in training and testing differ in their frequency of appearance.

Although we can assume that this is a fairly accurate prediction of movie reviews for a majority of cases, there is still scope for improvement in cases of false positive and false negative.