Nora Bockert
CS 506 Midterm
October 28, 2024

I began my approach by looking at X_trains dataset features to try and discern which features I thought were the most important. I believed that the Summary and Text were the most indicative features of the score, since they held people's thoughts about the movie. Initially, I thought SummaryLength and SummaryWordCount would be the most useful. I also believed that Time could be an important indicator to score because the time someone watched the movie and wrote the review could impact the score they rated it with. However, upon my first run, I realized that my accuracy was low and that these were not great features to look at. I did my research on how to best select features and found that there were packages that would help me extract the sentiment polarity from the ‘Summary’. In addition, I used CountVectorizer to extract keywords from ‘Summary’ and applied Latent Dirichlet Allocation to perform topic modeling on the words. I noticed that this process added a lot of extra columns to X_train and that when I would try to model the data, it would take a long time and produce low accuracy. After doing some more research, I discovered SelectKBest was an efficient way to select the best features and it would be more flexible to fit all datasets. I struggled and persevered with encoding and standardizing my data and to incorporate SelectKBest into my pipeline, but once it had been integrated, I experimented with values for k and n_neighbors. At first, I created a range and looped through the different values of n_neigbors to see which was most optimal. Then I discovered GridSearchCV, which was a more optimal solution and upon trial and error, I noticed that high values for n_neigbors and k were producing better accuracy, but my accuracy was around 57% which was not good enough. I played around with values in GridSearchCV and noticed that when I had the classifier_n_neighbors values at [90,100,110], 100 was returned as

the most optimal value and I realized I was overfitting the data past 100, so I settled on my parameters being [80,90,100]. I noticed that the dataset was unbalanced with a majority of 5 score ratings and I discovered that using SMOTE you could create synthetic values for minority values for the training set. However, upon implementing this it opened a whole world of problems. After SMOTE, I had to set n_neighbors values to be [2,3,4] and almost always, it would return 2 as the most optimal value. My code was taking longer to cross-validate the data and even though I was getting +70% accuracy on the training set, I was getting 44% accuracy on the testing set. Prior to this, I was testing on 10 percent of both test and training datasets and was using KNN with distance metrics, Euclidean and Manhattan and weights uniform and distance. But running with all of those candidates took 6 hours so I had to exclude Manhattan and uniform even though they were returning a higher accuracy. I also adjusted my cv score and all of this was improving my training set accuracy but my testing set was still super low. I figured out that the reason it was so low was because the feature selection was not being applied to the test_set. This was not a simple fix, I was up debugging until 2am. After finally applying feature selection to X_test, I was getting 76% accuracy on the training set and I got 26% accuracy on the testing set, so that's when I really went crazy. I traced how I handled X_test and made sure that I was not overfitting on my training data, but after all that debugging, the highest accuracy I could get was 44%. Additionally, in executing the entire datasets I removed grid search and set n_neighbors=90 because it was inefficient and time consuming. I adjusted my parameters to get the highest accuracy from the training set keeping k=50 and lowering my n_neighbors to testing 2 and 3. I decided to count my losses and remove SMOTE as it was the source of all my troubles and settle for an accuracy of 57%. Oh well.

Even though I would prefer having a midterm 1000%, I learned a lot about machine learning and data science practices that will be useful for my final project. It was very interesting to see how individual words were given sentiment polarity and subjectivity which became understandable to the computer after being standardized and encoded. In addition, the SMOTE tool is a really interesting feature in its ability to synthesize minority data, but I could not use it in a way that didn't result in it overfitting my training data.