**Introduction and Motivation** :

The Covid-19 pandemic has put everyone into quarantine which had led to more and more people on the internet and according to a recent survey “54% of internet users are watching more shows and films on streaming services due to Covid-19” (Statistica and Market.us, 2020). IMDb is an acronym for Internet Movie Database which is an online site to stream movies, tv shows, videos, games, etc. Movie reviews can play a major role in deciding whether or not a platform will remove/keep a movie from their site.

I am interested in researching the different accuracies of sentiment analysis with different machine learning algorithms. I wanted to see if I could find the algorithm that has the best accuracy of predicting a review's sentiment, positive or negative. If I can get an algorithm to be at least 80% accurate then I think it would be good enough to predict the sentiment for future movie reviews. Having a good enough algorithm can help save the company time from having to read each individual review and calculate the percentage of how many good or bad reviews there are.

The IMDb data set has 50,000 reviews which has a large enough dataset to generalize the algorithm to other movie reviews. If the dataset was only 10,000 reviews, then this may create some biases like: not having a diverse training set, can’t account for some words, or not a big enough training set, which I can highlight potential ways to solve them in the discussion section.

**Related Work**:

Now that everyone is at home due to the new stay at home order there are more and more people inclined to stream movies at home. This would be a better time than any to work on putting out the best content possible. I decided to find some research report on sentiment analysis and read through some research papers and found a couple I liked. The first literature I looked at was Sentiment Analysis of Movie Reviews using a Hybrid Method of Naive Bayes and Genetic Algorithm by M.Govindarajan . In this study, he took movie reviews from IMDb and tried two different methods to attain the best accuracy. He used Naive Bayes (NA), Genetic Algorithm (GA), and finally some hybrid combination of the two. This connects to the class course material since we went over the Naive Bayes Theorem. In the end he concluded that the hybrid algorithm had the best accuracy at 93.8%. The other two algorithms, NA and GA, sat at just above 91.1% which meant the new hybrid approach improved the accuracy by a whole 2 percent. I found this interesting because before reading this I was only thinking of trying different Machine Learning algorithms and then adjusting their parameters to see which has the best accuracy, now I will also try to find a way to combine two different algorithms.

The second research paper that I found useful was Sentiment Analysis of Movie Review Comments by Kuat Yessenov and Sasa Misailovic, two researchers at the Massachusetts Institute of Technology. They used movie reviews for Star Trek and a James Bond movie, from a social networking site called Digg.com to perform sentiment analysis by using Naive Bayes, Maximum Entropy, Decision Tree, and K Means Clustering. This also connects to the course material since it also used Naive Bayes and K means Clustering, both of which we had gone over. The analysis they did to categorize the polarity (positive vs negative) and subjectivity (objective vs subjective) of the movie reviews. I thought it was interesting how they used different variations of feature extraction for each algorithm. They used plain bag-of-words, bag-of-words using frequencies from the movie reviews corpus, and bag-of-words using only adjectives and adverbs and accounting for negation. This is a different way of feature extraction than mine since I used the TF-IDF that is associated with CountVectorization. In the end they found that Naive Bayes and Decision Tree with a plain bag of words perform almost equally well on both corpora at around 67% accuracy rate.

Both of the research papers that I read had Naive Bayes algorithms used although they had two very different accuracy ratings. The algorithms that I looked into were Support Vector Classification, Logistic Regression, and K Nearest Neighbors. All of these were mentioned in the introduction of the research papers.

**Methods:**

**Data preprocessing:** I downloaded my data from kaggle.com which were movie reviews from IMDb.com. I had to do some preprocessing steps in order to clean up the data. I started off by exploring the data. I looked at the dataset and it was (50000x2) which means it had 50,000 entries by 2 rows. The two columns were ‘review’ and ‘sentiment’ which would state whether it was positive or negative. I checked how many positive and how many negative reviews there were and it was split even 50/50. Then I checked how many words there were in each review by tokenizing each review and checking the length of that list. The depiction of Figure 1 below shows that there are about 1300 words per review. Figure 2 on the other hand shows the number of unique words per review, which is about 500. I took this into account to give me a good estimation as to where to start the number of features, a parameter for the CountVectorizer. I ended up setting features to 2000, 1000, 750, and 500. I made sure to include the standard english stop words, stop words is a set of words that don’t add meaning like the, a, she, he, to, etc. So that there will be less neutral data and more meaningful data to vectorize. Finally, I converted all of the positives/negatives entries into 1/0 and added it as a column into the data frame.

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

Figure 1 Figure 2

**Support Vector Classification:** This algorithm in general tries to find the line with the biggest margin to separate the data points. The larger the margin the smaller the error. If the margin is too small, then the data is too closely related and will have a lot of error and therefore a low accuracy. This algorithm comes in the sklearn package and takes in many parameters and the one that I mainly focused on was the kernel. This parameter took on each of the following: linear, poly, rbf, and sigmoid. Please note that this parameter is very dependent on how the data is spread across a graph. I had set all the other parameters to default in order to keep it consistent. I had trouble hypothesizing which kernel would yield the best accuracy since I couldn’t visualize the vectorization of words. Below in figure 3 is a data frame of all the accuracies with the different combinations of features for CountVectorizer and Kernel parameters.

Table

Description automatically generated

Figure 3

**Logistic Regression:** This second algorithm that I tried to find the best accuracies by finding the coefficients of a linear equation within a log function. The Logistic regression algorithm is a model based on the following equation, Text

Description automatically generated with medium confidence

 for which they take a maximum likelihood to find that the equation is still a linear combination The algorithm also has many parameters. The penalty parameter seems to specify the norm of the penalization. I passed in all of the penalty parameters which were: l1, l2, elasticnet, and none. The results of the accuracies are in the figure 4 below which shows that this algorithm did fairly well averaging over 80%.

Table

Description automatically generated

Figure 4

**K Nearest Neighbor:** The KNN algorithm determines the classification of a point, by combining the classification of the K nearest points. It’s supervised learning because it tries to classify a point based on the known classification of other points. I input the different algorithm parameters to obtain the following accuracies: auto, ball\_tree, kd\_tree, brute. I made sure to keep the other parameters set to default to mainly focus on how this parameter will affect the performance. In figure 5 we can see that this algorithm did not do so well with sentiment analysis since the average across the board was under 60%.

Table

Description automatically generated

Figure 5

**Discussion**:

Table

Description automatically generated

Figure 6

To conclude, I found that the top 3 best combinations of features and algorithms were

1. CountVectorizer(750 features) and SVC(kernel = “linear”)
2. CountVectorizer(2000 features) and LR(Penalty = “l2”)
3. CountVectorizer(2000 features) and LR(Penalty = “l1”)

Their accuracies were 82.4%, 81.9%, and 81.7%. I think these are good accuracies although they definitely can improve. One weakness that I just realized was the way I was splitting up my data. I did use a train test split in order to randomize the entries being used for the train or test sets, but this function doesn’t account for the number of positive/negative reviews. I could have used some function to pick 10% positive and 10% negative sentiment reviews in order for the test/train sets to be more evenly split. If the majority of the test set is negative reviews this will create a drop in accuracy for positive reviews which we can see in figure 6 above by the difference in precision and recall. Note that precision is the accuracy of predicted true positives whereas recall is the accuracy of predicted true negatives. One strength from my code is that I was able to explore a lot of different accuracies by changing the parameters. I think that my model may be up in the 90 percentile accuracy which would be good enough to predict future sentiment of movie reviews if it were to use the whole dataset instead of only 10% of it.

Some ideas for a future direction for improving the accuracy of these algorithms are maybe try using a different feature extraction like bag of words instead of the TF-IDF that is used for CountVectorization, only looking at adjectives and adverbs creating a code to recognize sarcasm, maybe creating a lexicon for slang words. I would recommend looking for studies that go in depth of finding a way to combine naive bayes with support vector classification. This is something I wanted to do in my project although I had run out of time. I will come back to this and find an implementation later.

Citation:

Govindarajan, M. *Sentiment Analysis of Movie Reviews Using Hybrid Method of Naive Bayes and Genetic Algorithm*. Dec. 2013, citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.643.1049&rep=rep1&type=pdf.

N, Lakshmipathi. “IMDB Dataset of 50K Movie Reviews.” *Kaggle*, 9 Mar. 2019, www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews.

“Online Video and Streaming Sites Statistics and Facts.” *Market.us*, 15 Sept. 2020, market.us/statistics/online-video-and-streaming-sites/.

Yessenov, K., & Misailovic, S. (2009, May 17). *Sentiment Analysis of Movie Review Comments* [Scholarly project]. Retrieved from http://people.csail.mit.edu/kuat/courses/6.863/report.pdf