Sana Khan

Homework 7 – MNB and SVN

Introduction

In today's age, it's easier than ever to watch a movie given the abundance of platforms and streaming services available to consumers. It's also easier than ever to write and read reviews that others have written about a movie. Bad reviews are the last thing any production company, or even actors would want to read about their performance. These reviews can impact future revenue, either positively or negatively.

One very popular website is Rotten Tomatoes, where critics and regular audience members can leave reviews. Critics will leave a written review and if the movie has at least 60% positive reviews a whole red tomato visual is applied. If less than 60% of reviews are positive, then a green splat tomato visual will be applied. Regular audience members also can leave written reviews and have a five-star rating system. These visuals are helpful as a consumer to get a good idea of how the movie is rated without having to read through reviews.

Being able to parse through reviews and analyze the sentiment is important in applying the correct score to a movie. If a review is positive but incorrectly categorized as negative, it could impact how many viewers will watch the movie, which could have an adverse effect on the movie's revenue. Getting the reviews sentiment right is a critical part not only for the website to be reputable, but also for the movie industry to generate interest.

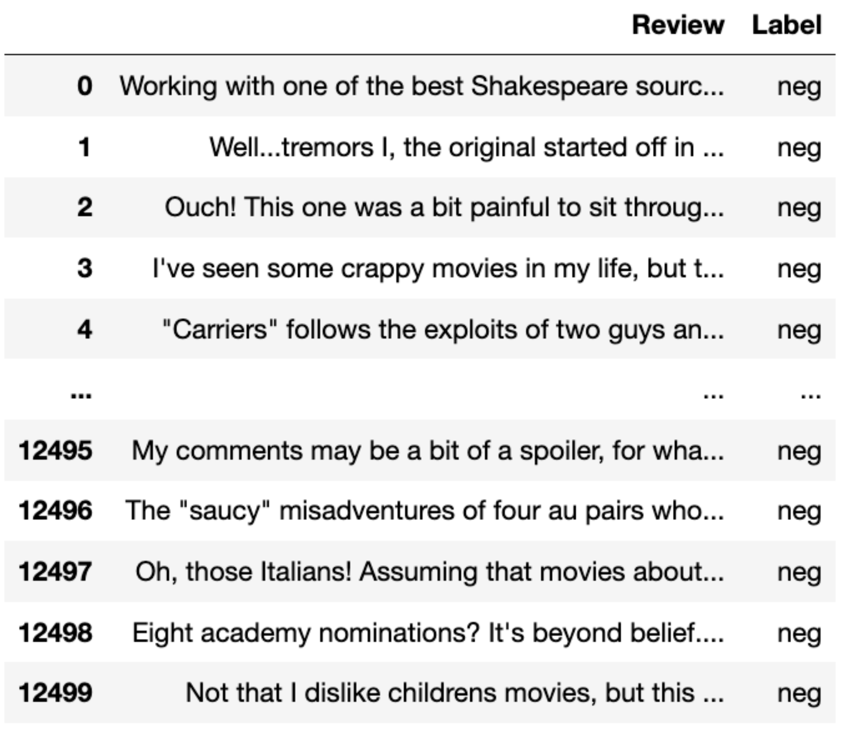
Our task is to use a subset of reviews to train our model to detect the sentiment of the review. We will then use the trained model on the rest of the reviews to understand how well the model was able to predict for each category. We will be using two models to determine which is better at predicting the sentiment of the reviews.

Analysis - Multinomial Naïve Bayes and SVM

Data Prep

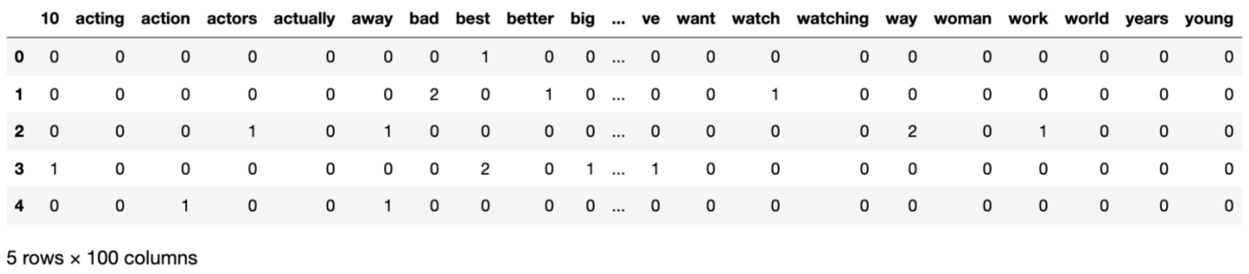
The data was downloaded from a google drive, where each sentiment, positive and negative, is in it's own folder. Each review was a separate text file, with 12,500 reviews per sentiment. The files were downloaded and each review was put into either a positive or negative folder depending on the sentiment. A function was then created that takes two parameters: a folder path a label. This function is designed iterate over the text files from the specified folder, extract the content of each file, and assign the label from the folder it was in. A data frame for the text files from each folder was created and both data frames were concatenated into one.

The resulting data frame has 25,000 rows and two columns. The new dataframe was then saved to a csv to be used later.



	Review	Label
0	Working with one of the best Shakespeare sourc...	neg
1	Well...tremors I, the original started off in ...	neg
2	Ouch! This one was a bit painful to sit throug...	neg
3	I've seen some crappy movies in my life, but t...	neg
4	"Carriers" follows the exploits of two guys an...	neg
...	...	...
12495	My comments may be a bit of a spoiler, for wha...	neg
12496	The "saucy" misadventures of four au pairs who...	neg
12497	Oh, those Italians! Assuming that movies about...	neg
12498	Eight academy nominations? It's beyond belief....	neg
12499	Not that I dislike childrens movies, but this ...	neg

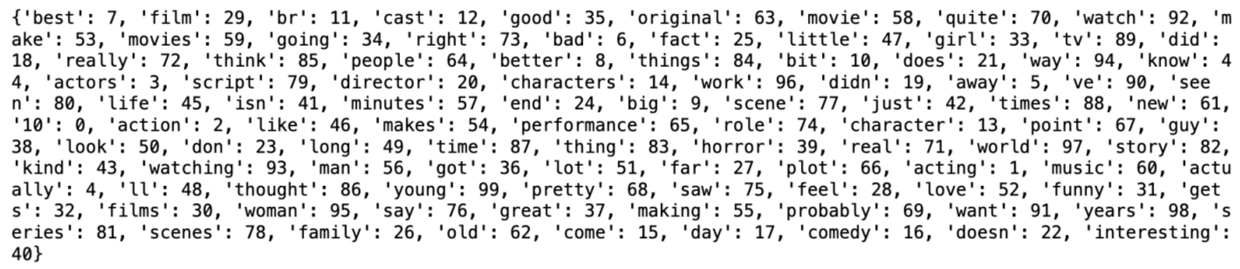
The csv was then read in, and each row was iterated over to create two new lists. One list is for the review and the other for the sentiment of the review. Then **CountVectorizer** from the scikit learn library was used to tokenize the words in the review list and put into a matrix. The matrix was then transformed to a data frame using pandas. Only the top 100 words were used as a parameter from CountVectorizer.



	10	acting	action	actors	actually	away	bad	best	better	big	...	ve	want	watch	watching	way	woman	work	world	years	young
0	0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	2	0	1	0	...	0	0	1	0	0	0	0	0	0	0
2	0	0	0	1	0	1	0	0	0	0	...	0	0	0	0	2	0	1	0	0	0
3	1	0	0	0	0	0	0	2	0	1	...	1	0	0	0	0	0	0	0	0	0
4	0	0	1	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 100 columns

The **get_feature_names_out** method was used to identify the column names in order to remove columns that contain numbers or are tokens that do not add value. These columns were removed using a function that identified numbers and a column with a length of less than two letters. The data was then split into testing and training datasets with a 70/30 split.



```
{'best': 7, 'film': 29, 'br': 11, 'cast': 12, 'good': 35, 'original': 63, 'movie': 58, 'quite': 70, 'watch': 92, 'm
ake': 53, 'movies': 59, 'going': 34, 'right': 73, 'bad': 6, 'fact': 25, 'little': 47, 'girl': 33, 'tv': 89, 'did':
18, 'really': 72, 'think': 85, 'people': 64, 'better': 8, 'things': 84, 'bit': 10, 'does': 21, 'way': 94, 'know': 4
4, 'actors': 3, 'script': 79, 'director': 20, 'characters': 14, 'work': 96, 'didn': 19, 'away': 5, 've': 90, 'see
n': 80, 'life': 45, 'isn': 41, 'minutes': 57, 'end': 24, 'big': 9, 'scene': 77, 'just': 42, 'times': 88, 'new': 61,
'10': 0, 'action': 2, 'like': 46, 'makes': 54, 'performance': 65, 'role': 74, 'character': 13, 'point': 67, 'guy':
38, 'look': 50, 'don': 23, 'long': 49, 'time': 87, 'thing': 83, 'horror': 39, 'real': 71, 'world': 97, 'story': 82,
'kind': 43, 'watching': 93, 'man': 56, 'got': 36, 'lot': 51, 'far': 27, 'plot': 66, 'acting': 1, 'music': 60, 'actu
ally': 4, 'll': 48, 'thought': 86, 'young': 99, 'pretty': 68, 'saw': 75, 'feel': 28, 'love': 52, 'funny': 31, 'get
s': 32, 'films': 30, 'woman': 95, 'say': 76, 'great': 37, 'making': 55, 'probably': 69, 'want': 91, 'years': 98, 's
eries': 81, 'scenes': 78, 'family': 26, 'old': 62, 'come': 15, 'day': 17, 'comedy': 16, 'doesn': 22, 'interesting':
40}
```

Model/Method

Two models were used to predict sentiment, Multinominal Naïve Bayes (MNB) and SVM. Since the dataset was labeled, this is a supervised learning model. Naïve Bayes works by taking the probability that a feature (word) will appear in a category. It does so by taking the prior probability of each feature and the likelihood of observing each feature given a category. It also makes the assumption that all features are independent of each other. SVM works very differently than MNB but still works on labeled data as a supervised learning model. SVM works by plotting the data based on the dimensions of the features, and aims to find the hyperplane that maximally separates the data points of different classes. This becomes the classifier, and once identified the model will subsequently classify new data points based on their position relative to this separator.

Once the model is trained on the training data using the fit method, it is applied to the testing data using the predict method. The testing data has the labels removed, since that is what the model is trying to predict. For MNB, it takes the probabilities from the testing data and uses that to predict which category the labels should appear in. For SVM, it utilizes the features of the testing data to determine the optimal hyperplane, enabling the prediction of the category to which the labels should be assigned. We then compare the data results with the labels that were removed from the dataset. Then a confusion matrix is used to access the model's performance. It reveals the number of true positives, true negatives, false positives, and false negatives for each dataset. For SVM, three different kernels were used to test the data. Linear SVM is used to classify a straight a straight line that separates the data into different classes. RBF is ideal for complex relationships and uses radial basis function, which measures the similarity between a data point and landmarks in the feature space. The Poly kernel uses polynomial functions to create decision boundaries of varying degrees.

Analysis

Multinomial Naïve Bayes

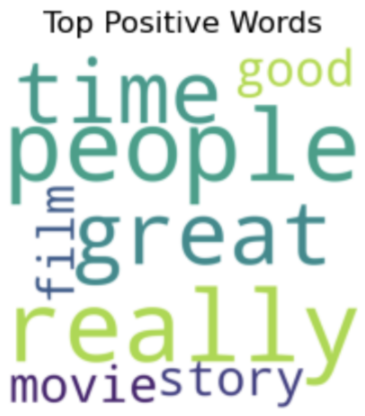
MNB was ran on the reviews list, and the following screenshot shows the probability for a subset of each review being either negative/positive. There are too many rows to show the full dataset, but because there is an even number of positive and negative reviews the model has an almost uniform chance of predicting either positive or negative.

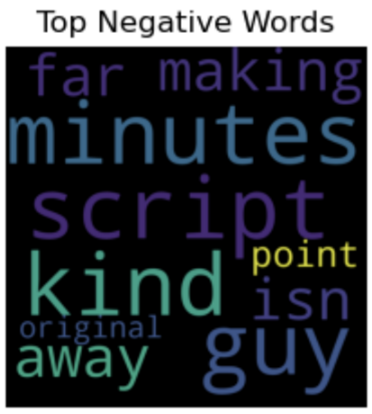
```
['pos' 'neg' 'pos' ... 'pos' 'pos' 'pos']
[[0.48 0.52]
 [0.67 0.33]
 [0.43 0.57]
 ...
 [0.15 0.85]
 [0.04 0.96]
 [0.42 0.58]]
```

This model did have an accuracy score of 71%. 2674 rows were correctly classified as negative. 1051 reviews were incorrectly classified as positive. 1108 reviews were incorrectly classified as negative. 2667 reviews were correctly classified as positive.

```
The confusion matrix
for the 100 word model is:
[[2674 1051]
 [1108 2667]]
```

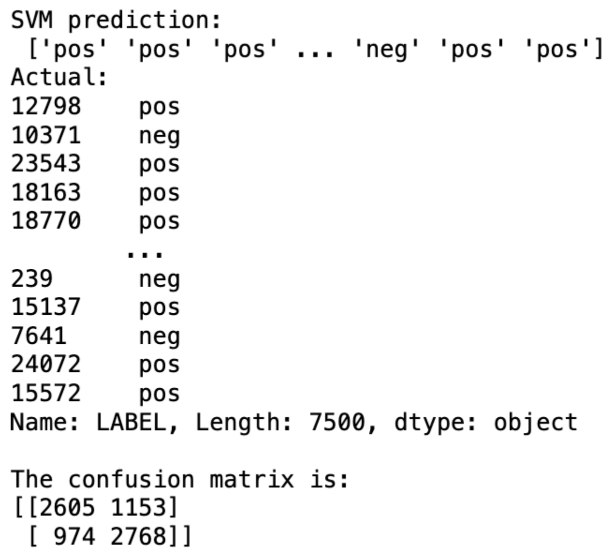
When plotting the word cloud, there aren't key words that identify the sentiment.





SVM

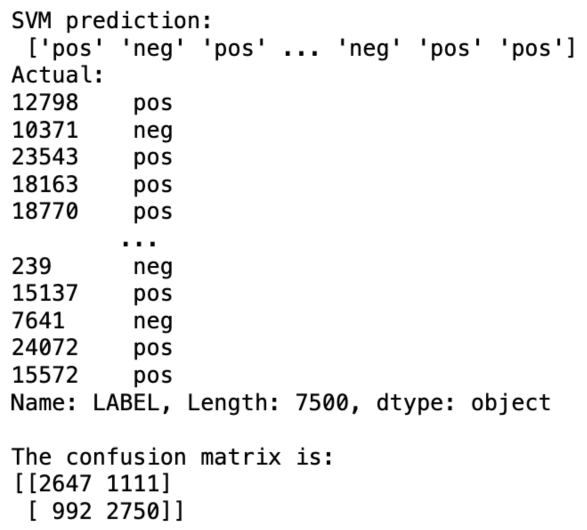
The SVM model was run three times, with the same training and test data split, with three different kernels all with a cost of one. The accuracy of the linear kernel model was 72%. The model correctly predicted 2599 reviews as negative, and 2773 reviews as positive.



```
SVM prediction:
 ['pos' 'pos' 'pos' ... 'neg' 'pos' 'pos']
Actual:
12798    pos
10371    neg
23543    pos
18163    pos
18770    pos
        ...
239      neg
15137    pos
7641     neg
24072    pos
15572    pos
Name: LABEL, Length: 7500, dtype: object

The confusion matrix is:
[[2605 1153]
 [ 974 2768]]
```

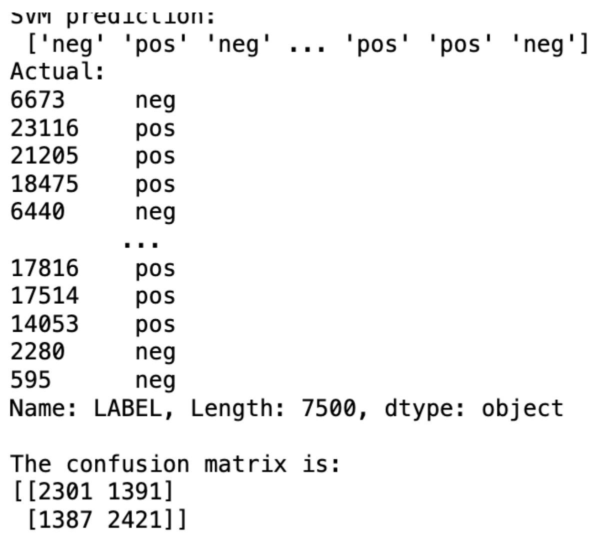
The False positive was predicted as 1159 reviews, and the false negative was 969 reviews. The model was run again with RBF as the kernel, and also had a 72% accuracy. The model accurately predicted 2647 reviews as negative, which is slightly higher than the linear kernel. The model also correctly predicted 2750 reviews as positive, which is slightly less than the kernel model. The model incorrectly predicted 1111 reviews as positive when they were actually negative and incorrectly predicted 992 reviews as negative when they were actually positive.



```
SVM prediction:
 ['pos' 'neg' 'pos' ... 'neg' 'pos' 'pos']
Actual:
12798    pos
10371    neg
23543    pos
18163    pos
18770    pos
        ...
239      neg
15137    pos
7641     neg
24072    pos
15572    pos
Name: LABEL, Length: 7500, dtype: object

The confusion matrix is:
[[2647 1111]
 [ 992 2750]]
```

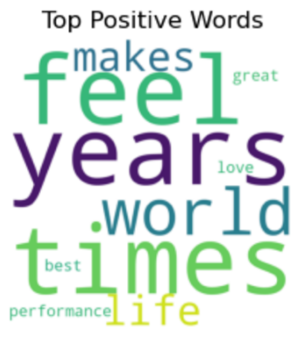
The third model had a kernel of poly, with the lowest accuracy score of 50%. This can be expected since this data is not the most ideal to use the poly kernel for, poly is better suited for image recognition tasks. The model accurately predicted 2301 reviews as negative and correctly predicted 2421 reviews as positive. The model incorrectly predicted 1391 reviews as positive when they were actually negative, and 1387 reviews as negative when they were actually positive.

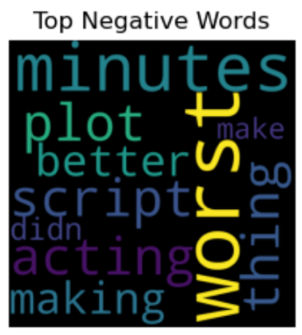


```
SVM prediction:
 ['neg' 'pos' 'neg' ... 'pos' 'pos' 'neg']
Actual:
6673     neg
23116    pos
21205    pos
18475    pos
6440     neg
        ...
17816    pos
17514    pos
14053    pos
2280     neg
595      neg
Name: LABEL, Length: 7500, dtype: object

The confusion matrix is:
[[2301 1391]
 [1387 2421]]
```

Taking a look at the word cloud, it's a bit better at identifying the positive/negative words as we can see more indicative words.





Conclusion

Both models had similar results in predicting the accuracy, around 70% for SVM and MNB. This data that was used may not have key words that indicate a strong positive or negative review, which can make it difficult to accurately predict results. To improve the accuracy, we probably should have increased our max word count, we have plenty of data to work from. We also may want to use other parameters in our model to correctly identify the sentiment of the review.