INTRODUCTION

The movie and film industry is definitely a very large one. The box office brings in approximately $42 billion a year in the last recent years. When adding in-home entertainment it passes $100 billion.

Many people enjoy watching movies for entertainment. Many of these people write reviews about the movies they watch. One could gain a lot of insight when reading some of these reviews.

What compels a movie to be reviewed as either negative or positive? Many filmmakers may embrace this question. Having knowledge of how a movie may be perceived as being a positive experience or a negative one can assist during the process of making a film, and possibly lead to the film having additional success.

This analysis will be a sentiment analysis on over 150 thousand movie reviews. We will build both a Multinomial Naïve Bayes classifier and a Support Vector Machine classifier to predict whether a review will be positive or negative. This analysis will also help to understand certain words that may lead to a review being positive or negative.

METHOD

For this method, we will first use a Naïve Bayes classifier, which is an algorithm that uses prior probabilities of certain events to predict future events. We will use a multinomial Naïve Bayes classifier, which is great for textual data, and deals with the probability of a word given a class with the frequency of the term in documents to the class.

We will then compare it to a Support Vector Machine (SVM) classifier. SVM algorithms classify data based on the cosine similarity of the data points that are aligned on the support vectors.

We will try both classifiers with a unigram feature set and a bigram feature set. We will start by splitting the data into 60% training and 40% testing, and then we will try the models with cross-validation to see if the results differ.

The first thing needed to do was to import the data into a pandas data frame, and then separate the phrases from the sentiment values. We then stored all of this two training variables.

After separating the values into X and y variables. We then split the data into a 60% training set and a 40% testing set. Here are the shapes of training and testing data sets as well as the first element of each:

```
(93636,) (93636,) (62424,) (62424,)
any new insight
2
a director enjoying himself immensely
3
```

The number values are the sentiment values. The values are 0 – 4, where 0 is very negative and 4 is very positive.

We then want to see how many reviews are in each of the sentiment labels:

```
[[    0  4231]
 [    1 16321]
 [    2 47693]
 [    3 19837]
 [    4  5554]]
```

By looking at these numbers, we can set a goal for our models. Since the category of '2' represents almost 51% of the data, we need a classifier that will be over 51% accurate to outclass what one could get by guessing that label.

We then developed some feature sets for how we were going to vectorize our data. We Python's NLTK package to use their stop words, but we also added some stop words of our own. We first want unigram features, which means only one word per value.

```
NLTKstopwords = set(stopwords.words('english'))
morestopwords = ['movie', 'film', 'story', 'just','would','could','like','
one','even']
stopwords = NLTKstopwords.union(morestopwords)
```

At first we attempted to add a regex pattern that excluded numbers, but that did not yield great results; so that was excluded.

We them proceeded to vectorize our review data.

Here is the shape of data and a quick view of an array of the first element. In addition, the length of the data's vocabulary, the first 10 words and the number of times it appears, as well as how many times the word amazing appears:

```
(93636, 12184)

[[0 0 0 ... 0 0 0]]

12184

[('new', 7228), ('insight', 5617), ('boy', 1330), ('meets', 6727),
('girl', 4629), ('posturing', 8134), ('good', 4701), ('chance', 1728),
('pass', 7711), ('stinker', 10284)]

455
```

We then transformed the vector to the testing data and started to build our Multinomial Naïve Bayes model. After just to check with human intuition if this was good, we took the log probability of the word 'amazing' to which label it belonged in:

```
-10.523633840211582 -10.686486862990368 -9.580634157072907 -
8.568876409773555 -7.724888439323074
```

The output means the feature 'amazing' is indicating that it is very positive because it holds the highest value amongst all conditional probabilities. By human intuition, this appears accurate.

We then repeated the model with a unigram and bigram feature set, which added combinations of one to two words.

We then repeated again, only using a Support Vector Machine model the same ways with a unigram feature set, and a unigram and bigram feature set. We proceeded to compare the results.

After we compared these different models, we then proceeded to build a Multinomial Naïve Bayes model and a Support Vector Machine model using cross-validation rather than splitting the data into a 60/40 split. We cross-validated using 10 folds, meaning we held out a different 10% of the data for testing 10 times, until we trained on all the data.

RESULTS

Multinomial Naïve Bayes: Unigram

We wanted to see the top ten words related to the very negative category. This was ranked by their log probabilities.

```
[(-6.166925013521991, 'rrb'), (-6.129184685539143, 'long'), (-
6.116914592947329, 'action'), (-6.080982583721266, 'time'), (-
6.057725721556999, 'dull'), (-6.057725721556999, 'worst'), (-
5.9489228617081995, 'characters'), (-5.841502613087362, 'minutes'), (-
5.778701711848332, 'comedy'), (-5.0388369067209275, 'bad')]
```

Then the same for the very positive category:

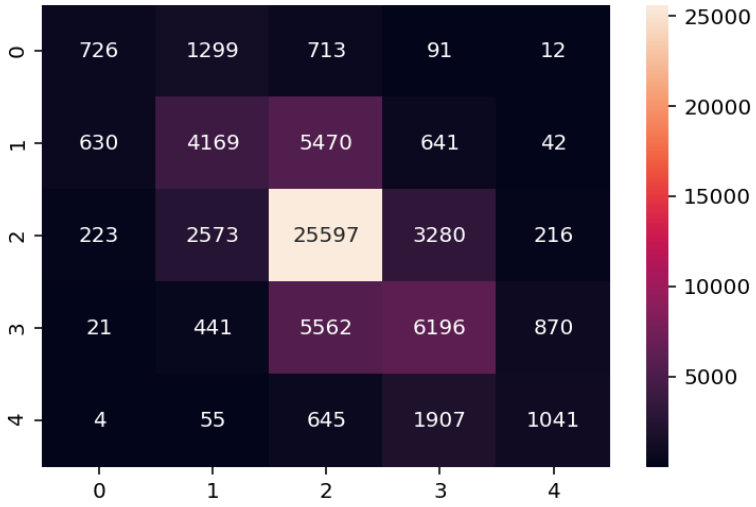
```
[(-5.9331289700950185, 'movies'), (-5.898642794023849, 'work'), (-
5.771487618538603, 'performance'), (-5.734853485358823, 'great'), (-
5.632374816075885, 'performances'), (-5.594153603255688, 'comedy'), (-
5.438218801634927, 'well'), (-5.406637229584629, 'good'), (-
5.366022510430438, 'funny'), (-5.172159193196312, 'best')]
```

Then we looked at the accuracy of the model, which was 60.4%

Here is the confusion matrix when predicting with the y testing set:

```
[[ 726 1299 713   91    12]
 [ 630 4169 5470  641   42]
 [ 223 2573 25597 3280  216]
 [ 21  441  5562  6196  870]
 [ 4   55   645   1907  1041]]
 precision recall f1-score support
0 0.45 0.26 0.33 2841
1 0.49 0.38 0.43 10952
2 0.67 0.80 0.73 31889
```

```
3 0.51 0.47 0.49 13090
4 0.48 0.29 0.36 3652
accuracy 0.60 62424 macro avg 0.52 0.44 0.47 62424 weighted avg 0.59 0.60
0.59 62424
```



Multinomial Naïve Bayes: bigram

Top ten words related to the very negative category.

```
[(-6.866973347761902, 'rrb'), (-6.829233019779054, 'long'), (-
6.81696292718724, 'action'), (-6.781030917961177, 'time'), (-
6.75777405579691, 'dull'), (-6.75777405579691, 'worst'), (-
6.648971195948111, 'characters'), (-6.5415509473272735, 'minutes'), (-
6.478750046088243, 'comedy'), (-5.738885240960839, 'bad')]
```

Top ten words related to the very positive category.

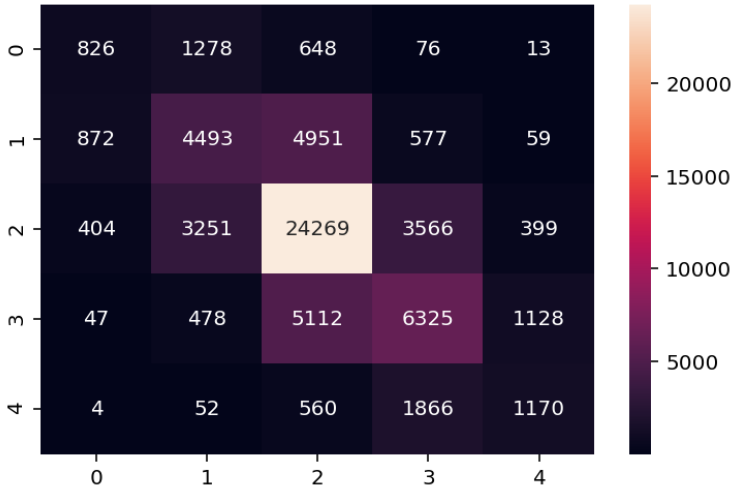
```
[(-6.591360829595552, 'movies'), (-6.556874653524383, 'work'), (-
6.429719478039137, 'performance'), (-6.393085344859356, 'great'), (-
6.290606675576418, 'performances'), (-6.252385462756221, 'comedy'), (-
6.096450661135461, 'well'), (-6.064869089085162, 'good'), (-
6.024254369930972, 'funny'), (-5.830391052696846, 'best')]
```

The accuracy was slightly lower than the unigram at 59.4%

Confusion matrix:

```
[[ 826 1278 648   76   13]
 [ 872 4493 4951  577  59]
 [ 404 3251 24269 3566 399]
 [ 47  478  5112  6325 1128]
 [ 4   52   560   1866 1170]]
precision recall f1-score support
0 0.38 0.29 0.33 2841
1 0.47 0.41 0.44 10952
2 0.68 0.76 0.72 31889
```

```
3 0.51 0.48 0.50 13090
4 0.42 0.32 0.36 3652
accuracy 0.59 62424 macro avg 0.49 0.45 0.47 62424 weighted avg 0.58 0.59
0.59 62424
```



Support Vector Machine: Unigram

Top ten words related to the very negative category.

```
(1.6509363195157987, 'loathsome') (1.653820897314448, 'ungainly')
(1.692678878445208, 'awfulness') (1.6943694045342914, 'disappointment')
(1.7135789140008713, 'grotesquely') (1.7144363284641098, 'atrocious')
(1.7928342716055066, 'worthless') (1.9403522101118431, 'unappealing')
(1.9792301572061186, 'sucked') (2.017241435466966, 'unwatchable')
```

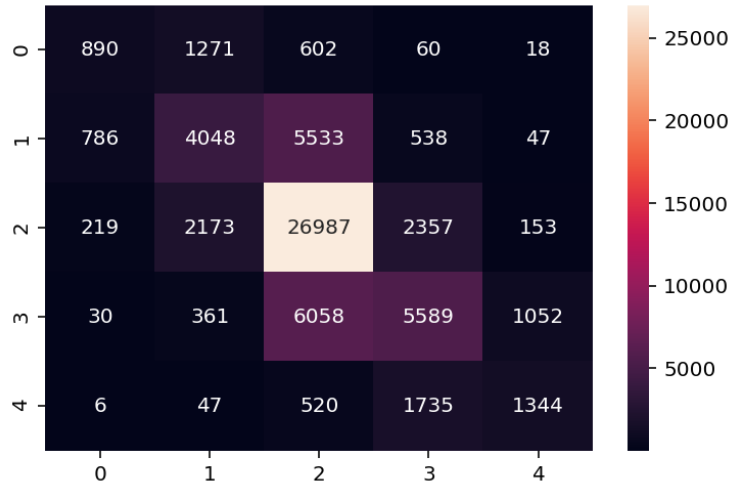
Top ten words related to the very positive category.

```
(1.5763240135177838, 'soars') (1.5821654643210308, 'glorious')
(1.5956713685039146, 'enriched') (1.5965183429134715, 'excellent')
(1.6413183867582988, 'masterfully') (1.6809237827533157, 'masterful')
(1.6901181271938857, 'flawless') (1.7133584721835164, 'magnificent')
(1.80309546192843, 'awesome') (2.013543715009282, 'perfection')
```

The accuracy was 62.2%

Confusion matrix:

```
[[ 726 1299 713   91   12]
 [ 630 4169 5470  641  42]
 [ 223 2573 25597 3280 216]
 [ 21  441  5562  6196 870]
 [ 4   55   645   1907 1041]]
precision recall f1-score support
0 0.46 0.31 0.37 2841
1 0.51 0.37 0.43 10952
2 0.68 0.85 0.75 31889
3 0.54 0.43 0.48 13090
4 0.51 0.37 0.43 3652
accuracy 0.62 62424 macro avg 0.54 0.46 0.49 62424 weighted avg 0.60 0.62
0.60 62424
```



Support Vector Machine: Bigram

Top ten features related to the very negative category.

```
[(1.7093359656948128, 'utterly incompetent'), (1.7181054832305875,
'grotesquely'), (1.7291806260626679, 'unappealing'), (1.7311666405621382,
'unbearable'), (1.7566730352415867, 'disappointment'),
(1.7771378476311939, 'thumbs'), (1.8145992988518376, 'unwatchable'),
(1.8220557170252676, 'appalling'), (1.887313644137342, 'sucked'),
(1.9094693701533045, 'worthless')]
```

Top ten features related to the very positive category.

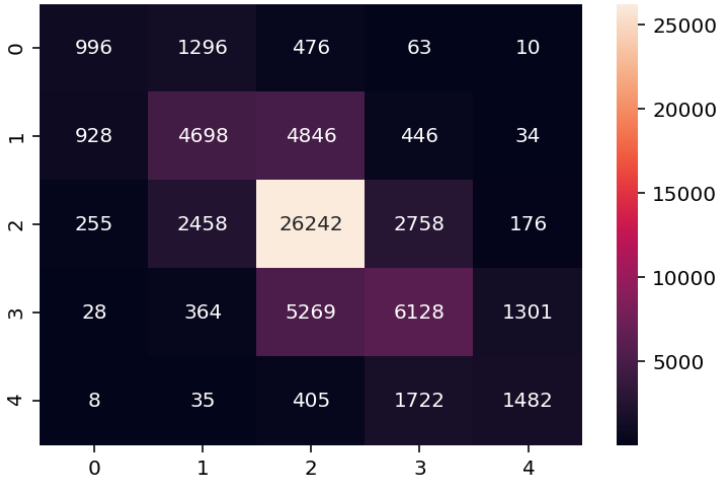
```
[(1.6736673040839765, 'superb'), (1.6809112110557356, 'masterful'),
(1.6849293531034593, 'flawless'), (1.7252649090064676, 'screenplay die'),
(1.7347776346682928, 'masterpiece'), (1.843956860148089, 'awesome'),
(1.8641953700479834, 'excellent'), (1.899679159203602, 'magnificent'),
```

```
(2.00736115537888, 'true heartbreaking'), (2.0080887964367466,
'perfection')]
```

The accuracy was slightly better than the unigram at 63.4%

Confusion matrix:

```
[[ 996 1296 476 63 10]
[ 928 4698 4846 446 34]
[ 255 2458 26242 2758 176]
[ 28 364 5269 6128 1301]
[ 8 35 405 1722 1482]]
precision recall f1-score support
0 0.45 0.35 0.39 2841
1 0.53 0.43 0.47 10952
2 0.70 0.82 0.76 31889
3 0.55 0.47 0.51 13090
4 0.49 0.41 0.45 3652
accuracy 0.63 62424 macro avg 0.55 0.50 0.52 62424 weighted avg 0.62 0.63
0.62 62424
```



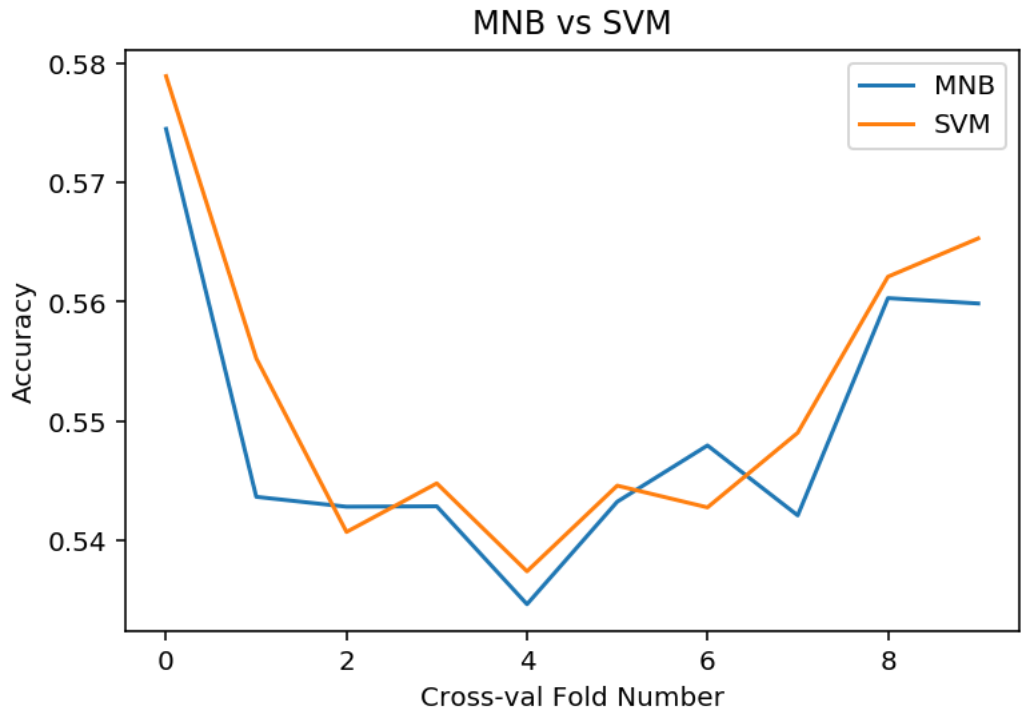
Multinomial Naïve Bayes: Cross Validation

With the same unigram feature sets, cross validating the data yielded us 54.9%

Support Vector Machine: Cross Validation

This was slightly better than the MNB. It was at 55.2%

Here is a graph that compares the accuracy of each model, at each fold of the cross-validation:



CONCLUSION

The model and feature set with the greatest accuracy was the SVM model with a bigram feature set. I do not believe that this was the best model though. I believe the best model was the SVM with cross-validation.

Although the accuracy was lower, cross-validation ensures that the training data is not biased by accident as it trains on all the data. A reason that splitting the data may have yielded higher results could be that part of the training data was biased.

In both models, it appeared the SVM performed better. Even though the numbers appear slight, when applied to a data set of around 100 thousand, that is a lot more predictions that are correct.

When looking at the top ten features for the very negative and very positive labels we can clearly see a difference in how the words are. When comparing models, The SVM definitely shows words that appear more influential. One thing a film maker may want to do with data like this is to possibly look at the top 100 words wo see if there are any features that have to do with how the film was made. The MNB model shows a couple of words relating to time, which could possibly translate to the length of a movie.

Although the movie industry may be lucrative, the filmmakers may fall in many pitfalls. Many movies lose money at the box office and an analysis of reviews could definitely assist when putting the film together.