

May 31, 2021

Quinn E Knudsen

IST 736

Syracuse University

Everyone’s a critic

Modeling Movie Review Sentiment at Scale

**Table of Contents**

**Introduction**……………………………………………………………………………......pg 2

Areas of Focus …………………………………………………………….……......pg 2

Movie Reviews..............……………………………………………………….........pg 2

Sentiment Analysis ………………………………………………………….…...... pg 3

**Analysis**……………………………………………………………………………….….…pg 4

About the Data ...……………………………………………………………….…...pg 4

Analysis 1: Sentiment Classification.……………………………………………..... pg 4

Data Cleaning & Prep..……………………………………………….….…..pg 4

**Results**…....………………………………………………………………………..………..pg 7

Analysis 1: Sentiment Classification ……………………………………….…...….pg 7

*Research Question 1………*………………………………………………....pg 7 *Model Summary…………………*………………………………………….....pg 15

**Conclusions**…………………………………………………………………………….…...pg 16

**Introduction**

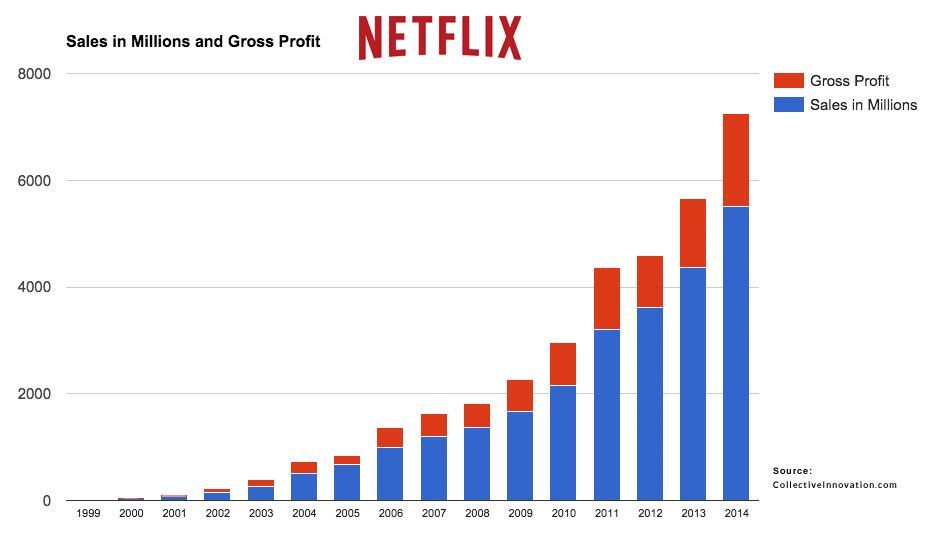
**Areas of Focus**

The scope of this paper is to explore classification problems utilizing a movie review dataset. Using Multinomial Naïve Bayes and Support Vector Machine (SVM) algorithms, various preprocessing a tuning techniques will be leveraged to predict movie review sentiment.

Through text analysis drawing from Python packages nltk. Finally, this paper will utilize Python packages such as sklearn and CountVectorizer to convert text corpuses into sparse matrixes for supervised machine learning classification. The objective of this research is to answer the question on whether or not machine learning algorithms can understand sentiment and whether or not the algorithm and preprocessing techniques impact model performance. Implications and future considerations will be discussed.

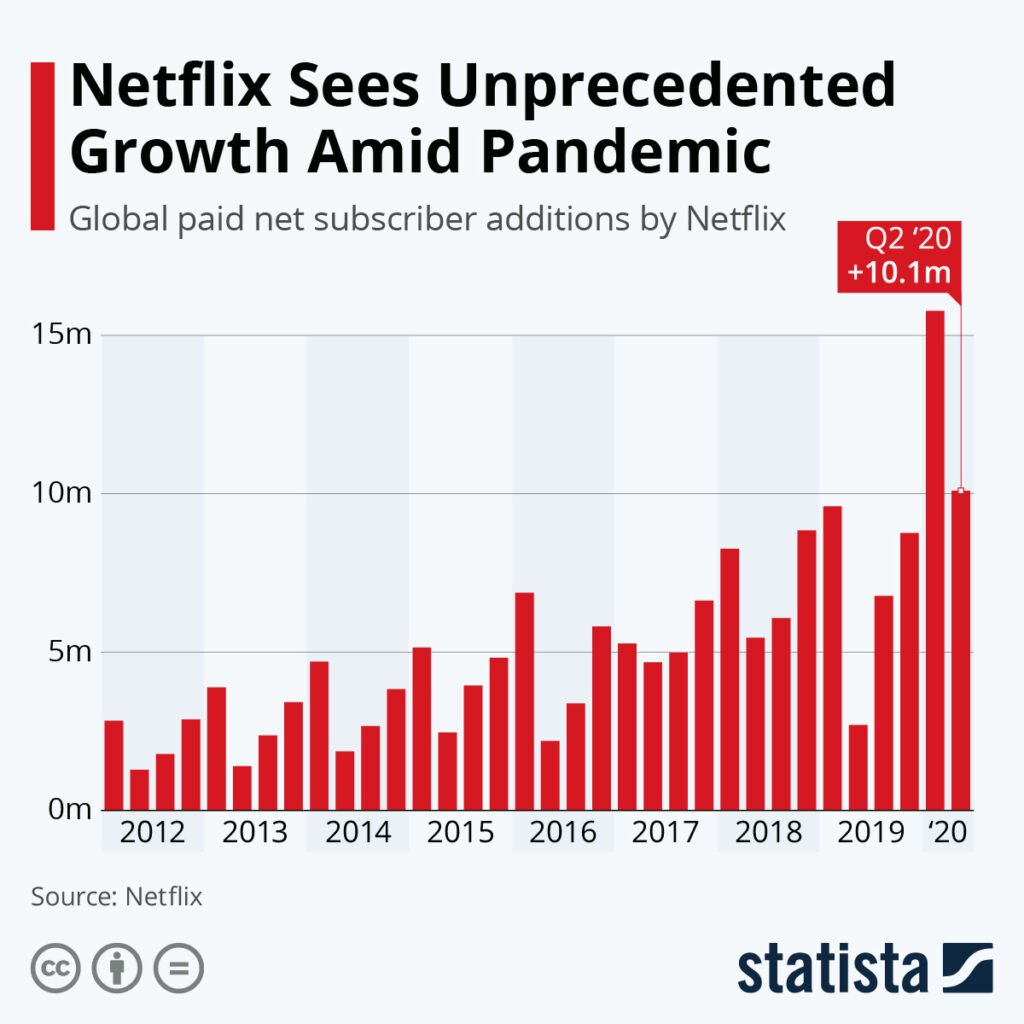
**Movie Reviews**

With the emergence of Netflix, Hulu, Amazon Prime and other stream services, it might be the wrong time to take a long-term investment position in Blockbuster. However, it is a great time to be a movie enthusiast with an all-time level of convenience and accessibility to one’s favorite shows, actors and directors. In fact, the growth of global video streaming such as amazing have grown exponentially as seen in Figure 1 below.



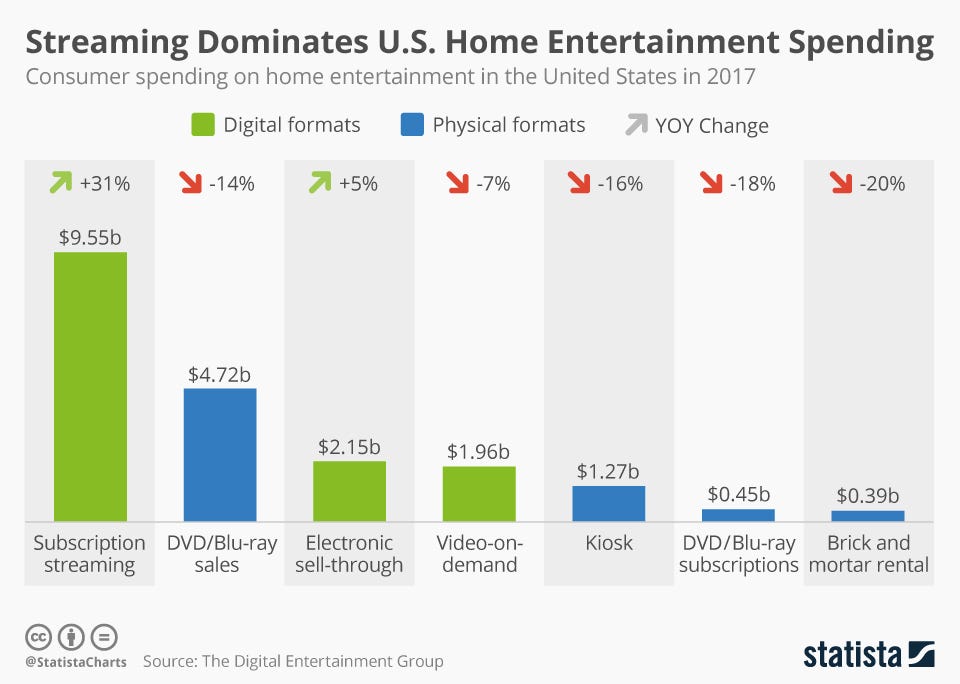
*Figure 1.* Growth of Netflix sales and gross profit from 1999-2014.

In recent years, amplified by the global pandemic, the demand for movie streaming services has only been amplified further. A growing number of traditional movie or television companies such as Disney and CBS now offer streaming subscription services online. Despite the growing market competition, Netflix remains the cream of the crop with market share and has continue to grow as noted in Figure 2.



*Figure 2.* Netflix dominance in streaming services has persisted in recent years.

Just as the movies have shifted over the years to digital providers, so have the manner in which movie goers seeks out ratings. Movie reviews are posted online and can even feed algorithms designed to suggest movies to online consumers. Scraping reviews for meaning can be an arduous but reward task. Companies that leverage traditional numeric ratings and also text analytics to understand the sentiment behind the movie watcher can create a competitive advantage in the marketplace.



*Figure 3.* Home entertainment spending by medium in the United States in 20117.

**Sentiment Analysis**

One applicant of machine learning popularized in recent years is text analytics-or the process of converting large amounts of unstructured written human linguistics into quantitative data capable of uncovering insights meaningful for human interpretation. Sentiment analysis is a specific domain of natural language processing that tries to determine whether the written information is positive, negative or neutral based upon the polarity of the words used in the text. This can prove to be a tall order due to the complexities of language including harder to detect nuances such as sarcasm, slang, negation or amplification. The partial focus of the analysis portion of this paper will explore an applicant of sentiment analysis on commentary related to movie reviews.

**Analysis**

**About the Data**

This analysis consisted of a dataset of movie reviews found on [Kaggle](https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data?select=train.tsv.zip). The data contained 156,060 rows and 5 columns. The columns include a phrase ID, sentence ID, a phrase (reviewing the movie) and a sentiment score ranging from 0-Negative to 4-Positive.

The sentiment labels are:

0 - negative  
1 - somewhat negative  
2 - neutral  
3 - somewhat positive  
4 – positive

|  |  |  |
| --- | --- | --- |
| **Category** | **Count** | **% of Total** |
| Negative | 7072 | 4.53% |
| Somewhat negative | 27273 | 17.48% |
| Neutral | 79582 | 50.99% |
| Somewhat positive | 32927 | 21.20% |
| Positive | 9206 | 5.90% |

*Table 1.* Class balance of the dataset shows bias towards neutrality and slightly more positive.

*Challenges*

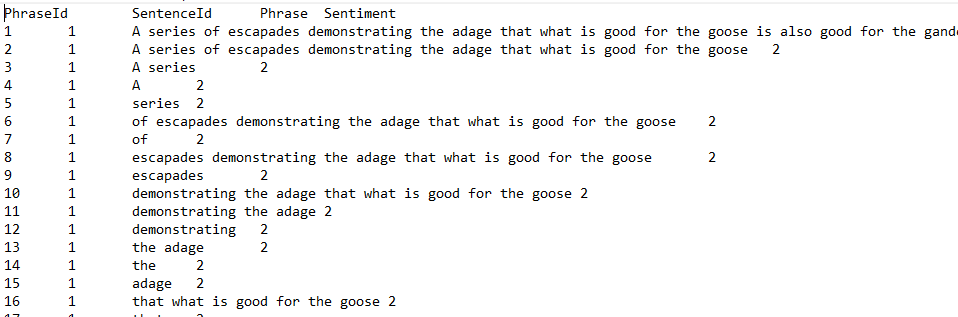
The occurrence of undesirable words (stop words) and alphanumeric characters needed to be removed. These would generally not be meaningful as columns in a sparse data frame and will need to be removed. These stop words were considered to be both the typical English words, numbers and select custom words. With greater customization comes a greater challenge in properly preparing the data frame. The final large-scale data preparation dilemma was how to deal with various iterations of words (capitalization, conjugation and other word syntax) that might unnecessarily widen the dataset or detract meaning from the analyses. Steps taken in the data cleaning did not utilize stemming or lemmatization, but future analyses would benefit from this approach when there are more revies, greater text and greater syntax requirements for analysis. All words were converted to lowercase for consistency.

**Data Cleaning & Prep**

This section will detail the steps taken to clean and prepare the datasets for further modeling using pictures to walk through the before during and aftereffects of the data preparation process.

*Before Prep*

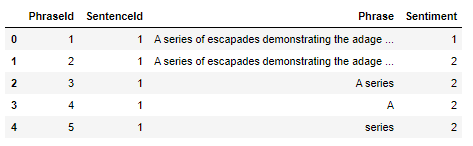
Prior to any preprocessing, as described, the dataset was not adequately prepared for conversion to a structured data frame. Before conversion to a document term matrix or tf-idf as will be detailed in the ensuing stages of preparation, the dataset appears as outlined in Figure 4.



*Figure 4.* Example extract of the raw data file.

*Reading in the Data and Initial Preparation*

Data was imported into Python as a tsv file and a data frame was created for the final clean version with columns as depicted in the Table 2 below.



*Table 2.* Simple labeled data frame.

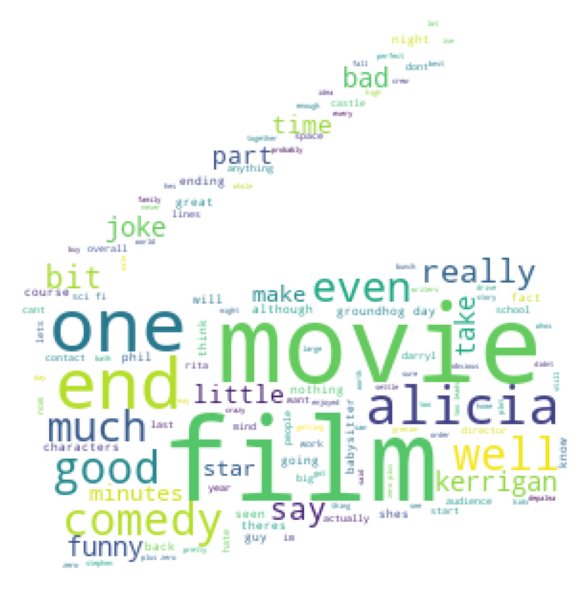
*Development of Word Clouds for Frequency*

Once again, leveraging the nltk stopwords and custom words “go” and “going”, the text was stripped of common English words and tokenized. Each tokenized word was converted to lowercase for consistency. These words were then visualized in a word cloud to show frequency as demonstrated in Figure 5 on the next page.



*Figure 5.* The most commonly used words in the movie reviews after removing stop words.

An additional visualization was created in the shape of a movie related image which could carry additional domain specific significance.



*Figure 6.* Domain specific representation of the most commonly used words in the movie reviews after removing stop words.

*Preparing Data for Document Term Matrix*

Next, labels were dropped from the file in order to make sure the data is ready to be converted into a document term matrix and labels should not be included in the matrix itself. This resulted in 5 elements in the list containing just the string data in row form.

*Document Term Matrix*

The unlabeled data was then passed into an empty list before CountVectorizer was used to transform the list based upon word occurrence. The array was then converted to a data frame as in analysis 1 and resulted in the following (see Table 3 for details).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Index** | **abcs** | **able** | **abound** | **absolutely** | **accordingly** | **action** | **activist** | **actual** |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

*Table 3.* Document term matrix for the movie data.

*Normalized Document Term Matrix (tf-idf)* *featuring English and Custom Stop Word Removal*

An additional step was taken to convert the document term matrix into a normalized document term matrix using tf-idf transformation. This transformation takes the normalized value for which the word occurs in the text matrix (proportionality). Common English stop words were removed from this sparse data frame along with words that are nontraditional English such as “abcs”. This conversion can be found in Table 4 below with final label results displayed in Table 9.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Index** | **abcs** | **able** | **abound** | **absolutely** | **accordingly** |
| 0 | 0 | 0 | 0.042975 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0.038633 | 0 |
| 3 | 0 | 0.047173 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0.059455 |

*Table 4.* TF-idf demonstrates the normalization and stop word removal.

**Results**

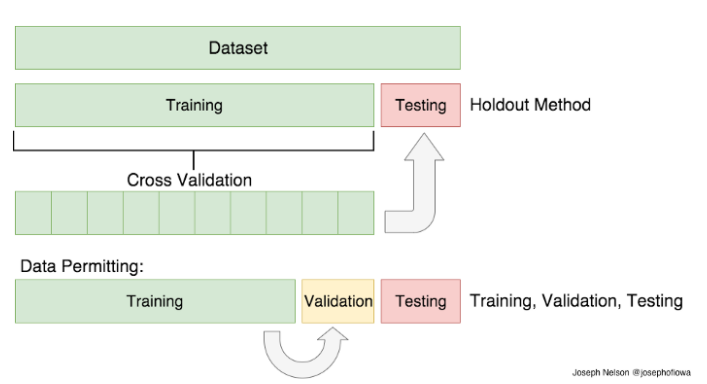
**Sentiment Classification of Movie Reviews**

*Research Question 1:* Can a model be trained to correctly predict sentiment based upon given movie review text?

In answering this question, the dataset was partitioned into train-test splits with 60% used for training and 40% for testing. Results did not go through a series of cross-validation due to runtime constraints.

*Train-Test Split*

For the classification modeling, the dataset was partitioned into training and testing. 66% of the data was used to train the model and 40% was held out to use for testing. As the sample was just 156,060 observations, this meant 93,636 observations were used to train and 62,424 were leftover to test. A visual representation of this can be observed in Figure 7 below. Splitting the data into train test splits helps determine the accuracy of the model and can be used to fine tune the approach (at the risk of overfitting).



*Figure 7.* Visual example of a train-test split with cross validation and testing.

**Models**

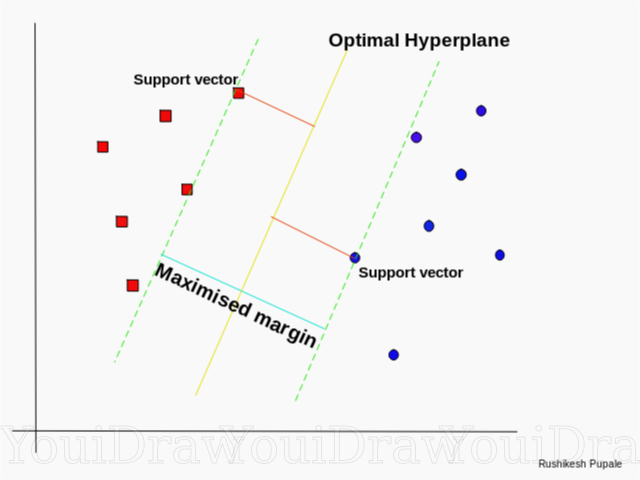
*Multinomial Naïve Bayes*

The Multinomial Naïve Bayes is a probabilistic algorithm often used in natural language processing based upon Bayes’ theorem with an assumption of independence in observations. In reality, it is understood that text violates this fundamental assumption, yet the usefulness of the algorithm is ascertained. This algorithm is useful for text analytics due to it’s ability to handle noise, missing values, can work quickly and scale on large datasets with parallel processing and it is straightforward in it’s interpretation. As this is a supervised method, it requires labeled data and in Python it requires numeric vectorized data inputs.

The multinomial naïve bayes algorithm was chosen for it’s ability to successfully process and classify sparse tdif (text-based) datasets.

*Supper Vector Machine (SVM)*

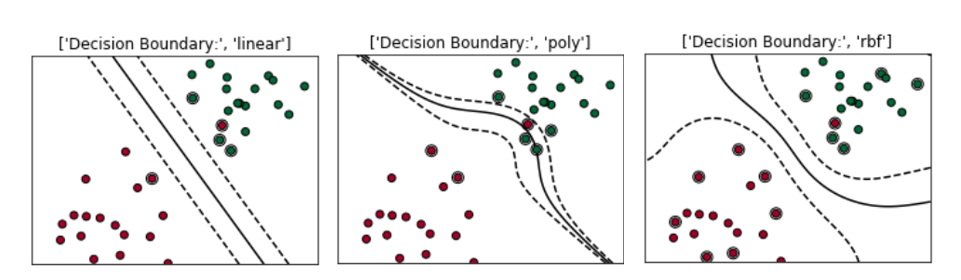
The support vector machine is machine learning algorithm that uses mathematical hyperplanes to optimize a classification problem derived from numeric variables. The goal of the algorithm is to maximize the distance of a mathematical line used to separate the data into distinct categories in n-dimensional space. A visual representation of this concept can be found in Figure 8.



*Figure 8.* Visual representation of an SVM margin classification.

*Kernels*

Although many SVM kernels exist, this paper utilizes three distinct kernel functions: linear, polynomial and radial. Of an infinite number of potential solutions, the linear classifier maximizes the distance between points with a formulaic straight line. A polynomial SVM typically uses a quadratic degree to draw a nonlinear line. The radial SVM is one of the most popular kernel calls and operates as the default in sklearn and often can maximize the classification with highly dimensional data. As seen in Figure 9 below, the manner in which the kernel operate can differentiate the classification accuracy based upon the fit of the line “drawn”.



*Figure 9.* Visual example SVM kernel classification.

The models were preprocessed using the CountVectorizer and tf-idf techniques mentioned earlier. Each model was tested at least once with each preprocessing methodology. In the follow two sections, the model confusion matrix results are displayed.

**CountVectorizer Models**

*Confusion Matrices*

The ensuing confusion matrices depict model results with a table of correct classification and a demonstration of precision, recall, f1-score and support (number of classifications for the level). These metrics are often more usual than accuracy when there is class imbalance in a dataset like the one used for this paper. If this model predicted every outcomes as neutral it would be ~51% accurate. A useful model with supersede this random guessing or no information rate.

*Precision*

Precision measures the number of true positives (positives correctly predicted as positive) divided by the number of positive predictions made by the model. This calculation is articulated below. Precision answers the question: “what proportion of positive identifications were actually correct?”.

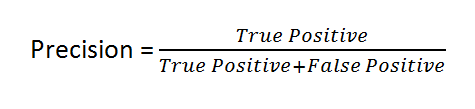


Figure 10. Mathematic formula for calculation of precision.

*Recall*

Recall measures the number of true positives (positives correctly predicted as positive) divided by the number of actual positive outcomes in the dataset. This calculation is articulated below. Recall answers the question: “what proportion of actual positives were identified correctly?”.

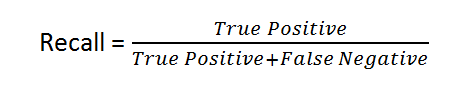


Figure 11. Mathematic formula for calculation of recall.

*F1 Score*

F1 score measures the accuracy of the model by combining the precision and recall to get a harmonic mean of the two classification indicators. This accounts for the ‘correctness’ of the model relative to the number of true positive and false positives as articulated in the mathematical equation below. F1-score was used in this paper due to the goal to maximize both precision and recall.

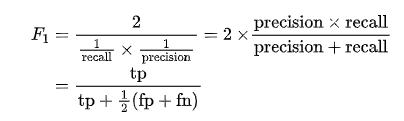


Figure 12. Mathematic formula for calculation of the F1-Score.

**SVM Linear CountVectorizer**

[[ 918 1221 697 82 13]

[ 701 4080 5504 514 25]

[ 195 2106 27081 2310 172]

[ 34 396 6048 5533 1057]

[ 3 51 590 1772 1321]]

precision recall f1-score support

0 0.50 0.31 0.38 2931

1 0.52 0.38 0.44 10824

2 0.68 0.85 0.75 31864

3 0.54 0.42 0.48 13068

4 0.51 0.35 0.42 3737

accuracy 0.62 62424

macro avg 0.55 0.46 0.49 62424

weighted avg 0.60 0.62 0.60 62424

**SVM Radial CountVectorizer**

[[ 696 1506 662 64 3]

[ 417 4358 5640 400 9]

[ 84 1785 27806 2141 48]

[ 5 242 5876 6404 541]

[ 0 25 574 2191 947]]

precision recall f1-score support

0 0.58 0.24 0.34 2931

1 0.55 0.40 0.47 10824

2 0.69 0.87 0.77 31864

3 0.57 0.49 0.53 13068

4 0.61 0.25 0.36 3737

accuracy 0.64 62424

macro avg 0.60 0.45 0.49 62424

**Multinomial NB CountVectorizer**

[[ 694 1493 682 58 4]

[ 391 4505 5554 360 14]

[ 83 1978 27547 2187 69]

[ 8 259 5808 6424 569]

[ 0 27 554 2143 1013]]

precision recall f1-score support

0 0.59 0.24 0.34 2931

1 0.55 0.42 0.47 10824

2 0.69 0.86 0.77 31864

3 0.58 0.49 0.53 13068

4 0.61 0.27 0.37 3737

accuracy 0.64 62424

macro avg 0.60 0.46 0.50 62424

weighted avg 0.63 0.64 0.62 62424

**SVM Poly CountVectorizer**

[[ 759 1258 871 39 4]

[ 469 4219 5861 265 10]

[ 110 1985 27411 2271 87]

[ 11 213 6319 5878 647]

[ 1 21 809 1892 1014]]

precision recall f1-score support

0 0.56 0.26 0.35 2931

1 0.55 0.39 0.46 10824

2 0.66 0.86 0.75 31864

3 0.57 0.45 0.50 13068

4 0.58 0.27 0.37 3737

accuracy 0.63 62424

macro avg 0.58 0.45 0.49 62424

weighted avg 0.61 0.63 0.61 62424

**TfidfVectorizer Models**

*Confusion Matrices*

**SVM Linear tf-idf**

weighted avg 0.63 0.64 0.62 62424

[[ 795 1387 624 117 8]

[ 589 4336 5245 629 25]

[ 163 2299 26557 2684 161]

[ 24 408 5604 6220 812]

[ 2 40 551 2010 1134]]

precision recall f1-score support

0 0.51 0.27 0.35 2931

1 0.51 0.40 0.45 10824

2 0.69 0.83 0.75 31864

3 0.53 0.48 0.50 13068

4 0.53 0.30 0.39 3737

accuracy 0.63 62424

macro avg 0.55 0.46 0.49 62424

weighted avg 0.61 0.63 0.61 62424

**SVM Radial tf-idf**

[[ 694 1493 682 58 4]

[ 391 4505 5554 360 14]

[ 83 1978 27547 2187 69]

[ 8 259 5808 6424 569]

[ 0 27 554 2143 1013]]

precision recall f1-score support

0 0.59 0.24 0.34 2931

1 0.55 0.42 0.47 10824

2 0.69 0.86 0.77 31864

3 0.58 0.49 0.53 13068

4 0.61 0.27 0.37 3737

accuracy 0.64 62424

macro avg 0.60 0.46 0.50 62424

weighted avg 0.63 0.64 0.62 62424

**Multinomial NB ti-idf**

[[ 694 1493 682 58 4]

[ 391 4505 5554 360 14]

[ 83 1978 27547 2187 69]

[ 8 259 5808 6424 569]

[ 0 27 554 2143 1013]]

precision recall f1-score support

0 0.59 0.24 0.34 2931

1 0.55 0.42 0.47 10824

2 0.69 0.86 0.77 31864

3 0.58 0.49 0.53 13068

4 0.61 0.27 0.37 3737

accuracy 0.64 62424

macro avg 0.60 0.46 0.50 62424

weighted avg 0.63 0.64 0.62 62424

**SVM Poly tf-idf**

[[ 759 1258 871 39 4]

[ 469 4219 5861 265 10]

[ 110 1985 27411 2271 87]

[ 11 213 6319 5878 647]

[ 1 21 809 1892 1014]]

precision recall f1-score support

0 0.56 0.26 0.35 2931

1 0.55 0.39 0.46 10824

2 0.66 0.86 0.75 31864

3 0.57 0.45 0.50 13068

4 0.58 0.27 0.37 3737

accuracy 0.63 62424

macro avg 0.58 0.45 0.49 62424

weighted avg 0.61 0.63 0.61 62424

**Model Summary**

As articulated in Table 5 below, there are tradeoffs to be made with model complexity, efficiency and accuracy. For example, the highest scoring models (SVM RBF) scored ~65% accurate in classifying sentiment. However, they also took nearly 90 minutes of train time to build and this computational constraint should be considered when choosing the modeling technique for future projects.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Prep** | **Accuracy** | **Efficiency** |
| SVM C=1.5 RBF | Tf-idf | 0.646706 | Low |
| SVM RBF | CountVectorizer | 0.644159 | Low |
| SVM RBF | Tf-idf | 0.643711 | Low |
| SVM Polynomial | Tf-idf | 0.629261 | Low |
| SVM Linear | Tf-idf | 0.625433 | Moderate |
| SVM Linear | CountVectorizer | 0.623686 | Moderate |
| SVM C=5 Linear | CountVectorizer | 0.619617 | High |
| SVM C=100 Linear | CountVectorizer | 0.615853 | High |
| Multinomial NB | CountVectorizer | 0.606401 | High |
| SVM Polynomial | CountVectorizer | 0.600378 | Low |
| Multinomial NB | Tf-idf | 0.583606 | High |

*Table 5.* Model results demonstrate tradeoffs in efficiency and model performance.

A seemingly reasonable tradeoff can be made in accuracy, speed (and inherently complexity). While the Multinomial Naïve Bayes was the fastest to run, the combination of slightly more accuracy with marginally more train time made the SVM Linear classifier a standout model choice. Interestingly, the preprocessing of CountVectorizer and tf-idf made little difference and the CountVectorizer outperformed the tf-idf contrary to expectations. Additional REGEX and lemmatization would also provide avenue for increasing model accuracy through data prep.

The results of this model (found in Table 6) demonstrate that understanding neutral sentiment was the easiest to classify. It also demonstrated a slight bias towards being more accurate in classifying positive statements over negative statements. This could indicate that negation should be considered, and this would enhance the model performance. This also serves as a limitation for this model as movie goers and critics would like prefer to understand polarity (really poor reviews and really positive reviews). The classification could be enhanced with tuning with that consideration and the numbers of responses (Positive 🡪 Neutral 🡪 Negative) reduced to perhaps increase accuracy and practical utilization.

|  |  |
| --- | --- |
| **Category** | **Avg F-1 Score** |
| Negative | 0.35 |
| Somewhat negative | 0.46 |
| Neutral | 0.76 |
| Somewhat positive | 0.52 |
| Positive | 0.38 |

*Table 6.* Average F-1 scores across all models.

**Conclusion**

As Benjamin Franklin aptly put, “any fool can criticize, condemn and complain and most fools do”. While just a handful of professionals exist in the world, there are few mediums in which average patrons characterize themselves as an expert worthy of providing critical feedback on another person’s work like the movie industry.

Despite the lack of credentials and training, these everyday critics can really drive and shape the movie industry through online reviews. The emergence of machine learning curated content in streaming channels and the surge of online platforms that ask for movie reviews has resulted in the datafication of the movie industry with thousands of quantitative and qualitative movie reviews at the disposal of the would be patron’s disposal when making a choice for their next Netflix and Chill night in.

This paper found evidence that a model can be trained to understand human text and correctly categorize it with fair accuracy into specific levels of sentiment. Models are challenged in deciphering whether or not a statement is negative or somewhat negative and positive and somewhat positive. For practical utilization, additional work can be done to help the model make it’s decision including collapsing the categories into one two options: positive or negative.

Ultimately, movie review sentiment classification can save countless hours of work and add additional information for the consumer, provider and film-making industry to more deeply understand what makes a movie successful or unsuccessful and should be considered as a viable element of the film curation and creation process.