# Sentiment Analysis For IMDB Movie Reviews

By
Archana Kalburgi (10469491)
Arya Guddemane Vishwakumar (10459529)

## Abstract

Movies are rated in many ways, star-based ratings, number-based ratings, percent-based ratings, etc.

All these inform the viewers about the movie. Movie reviews expressed in public forums could also act as a reliable source of the quality of a movie. In addition to critic rating. In this project, we are exploring ways to identify sentiments expressed in written text.

Sentiment Analysis is tied to natural language processing and text mining. Sentiment Analysis aims to extract subjective information from the corpus of the text. Using sentiment analysis we can infer the state of the mind of the reviewer.

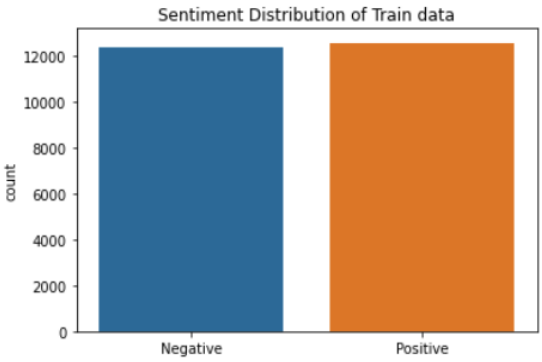
In this project, we use sentiment analysis on a set of movie reviews and try to infer their opinion of the movie. We have obtained predictions without feature extractions and after extractions. We also have tabulated the performance of various machine learning algorithms. We have focused on algorithms that we learned during the coursework. The algorithms we have discussed are Support Vector Machine, K-nearest neighbor, Random Forest, and Logistic Regression. And also used principal components analysis to reduce the dimensions of the dataset

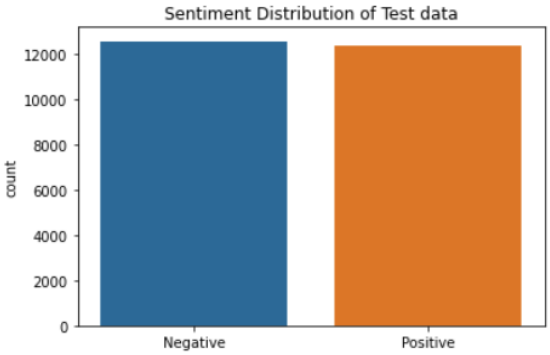
## Dataset Information

The dataset used in this project is the one for binary sentiment classification containing 25,000 highly polar reviews for training and 25,000 for testing.

## Data Exploration

To get a better insight into the data used in this project, we have made various plots. To begin with, the following histogram bar plots show the distribution of positive and negative sentiments in both the train and test set.



## Machine Learning Model

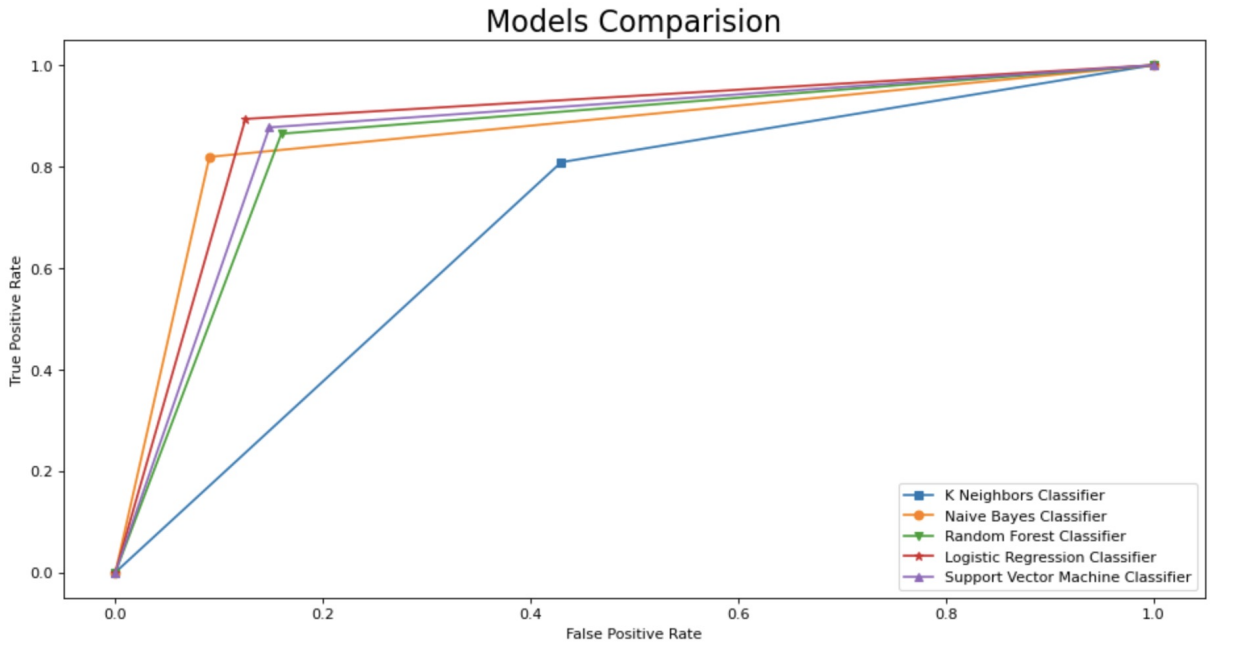
Models used in the project:

1. Support vector machine
2. Naive Bayes
3. K-Nearest Neighbours
4. Random Forest
5. Logistic Regression

The raw data was split into 50% train set and remaining 50% as test set, to apply all the above machine learning algorithms. Following table tabulates the accuracies of all the models

| Algorithms | Accuracies |
|---|---|
| Support Vector Machine | 0.864 |
| Naive Bayes | 0.866 |
| KNN | 0.685 |
| Random Forest | 0.852 |
| Logistic Regression | 0.884 |

Following is the ROC plot of all the algorithms:



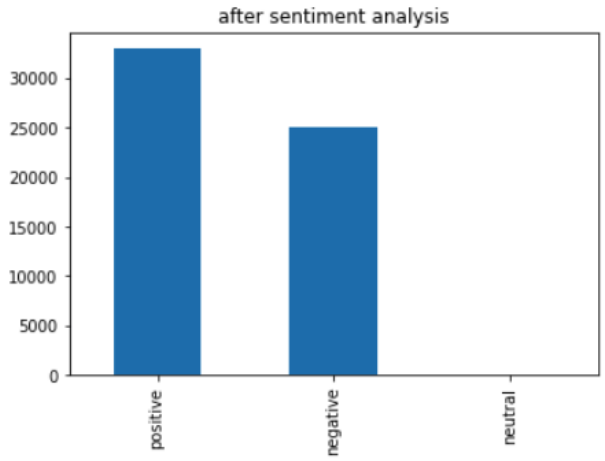
## Why Feature Extraction?

Using the SentimentIntensityAnalyzer and polarity_scores method we obtained the scores of the positive, negative, neutral and compound of all the "reviews" in the data. Compound score is the sum of positive, negative and neutral scores which is then normalised between -1(negative) and +1(positive). We did this step to compare the VADER's sentiment prediction and the actual label given in the data. The difference can be seen in the following histogram bar plot.

The difference between the VADER score and the actual sentiment labels prompted us to extract the features from the data.

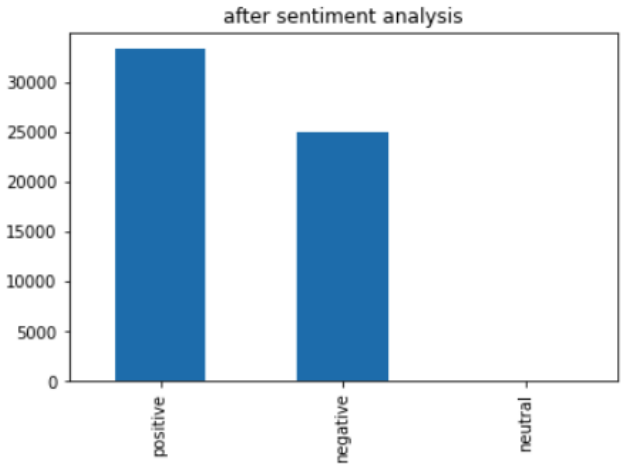
Vader polarity score raw data :

15419 -> differences between actual label and Vader predicted label




Vader polarity score after removing non-words or stop words and stripping html tags

15248 -> differences between actual label and Vader predicted label




## Cleaning

Cleaning the data and removing the redundant information is beneficial in machine learning projects. In this project we have applied multiple cleaning procedures on the data. Before any cleaning procedure was applied the most common words in our data can be seen in the following plot:


To get rid of some strange letters like ”br” which was the most repetitive one, we began removing HTML tags like “<br>”. We have used BeautifulSoup to remove these HTML tags from the text.

After cleaning, we proceeded to explore which were the common words in the data. The following Figure1 shows the most frequent words in the data.

As we can see the most frequent words were “the”, “and”, “a”, “of" and so on, which do not really convey any emotions when associated with other words in the review. Getting rid of such redundant words was our next task. With the help of nltk library that has a corpus of stopwords, we were able to remove these stopwords from the data. The common words after the removal of the stopwords were as shown in Figure2.

| | common_words | count |
|---|---|---|
| 0 | the | 39612 |
| 1 | and | 21601 |
| 2 | a | 21416 |
| 3 | of | 19693 |
| 4 | to | 18006 |
| 5 | is | 14739 |
| 6 | in | 12024 |
| 7 | I | 11409 |
| 8 | it | 10650 |
| 9 | that | 9331 |
| 10 | this | 8526 |
| 11 | s | 8324 |
| 12 | was | 6460 |
| 13 | The | 6171 |
| 14 | movie | 6145 |
| 15 | as | 5869 |
| 16 | with | 5852 |
| 17 | for | 5602 |
| 18 | film | 5484 |
| 19 | on | 4705 |

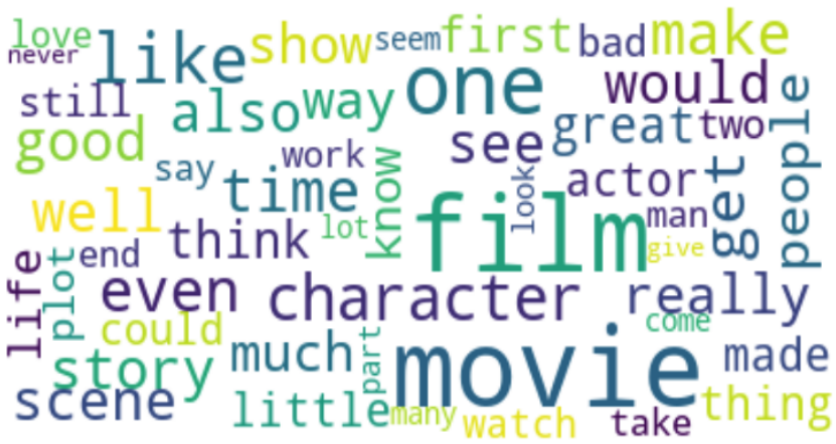
Figure1: Words before the stopwords were removed

| | common_words | count |
|---|---|---|
| 0 | movie | 59088 |
| 1 | film | 54201 |
| 2 | one | 33582 |
| 3 | like | 24748 |
| 4 | good | 19042 |
| 5 | time | 17051 |
| 6 | would | 15778 |
| 7 | story | 15663 |
| 8 | see | 15494 |
| 9 | really | 15223 |
| 10 | even | 14516 |
| 11 | well | 13065 |
| 12 | great | 12796 |
| 13 | much | 12305 |
| 14 | get | 12162 |
| 15 | bad | 12023 |
| 16 | people | 11829 |
| 17 | also | 11025 |
| 18 | first | 10914 |
| 19 | made | 10758 |

Figure 2: Words after the stopwords were removed

Along with removal of the stopwords we have made the data insensitive to case, which makes “The" and “the” the same word and not count as two different words.

Once the raw data was cleaned the clean data looked as follows. Which represent the most frequent words.




Feature Extraction

We have normalised the data using z-score standardisation which for some values returns a negatively scaled data. Naive Bayes algorithms cannot handle such negative data which led us to drop this algorithm for further exploration.

After cleaning the data the number of positive words and negative words in each reviews were as follows:

Using CountVectorizer(), we have converted the collection of text documents to a matrix of token counts that enabled us to represent the data in numeric format.

The list of features extracted from the clean data are as follows:

| Feature | Explanation | Example |
|---|---|---|
| neg | VADER's negative score | - |
| neu | VADER's neutral score | - |
| pos | VADER's positive score | - |
| 'compound' | VADER's negative score | - |
| length_clean | length of the cleaned review | - |
| count_words_clean | number of words in cleanied review | - |
| count_words_stop_words | number of stop words in the review | - |
| length_stop_words | length of the stop words | - |

| | | |
|---|---|---|
| PUNC_EXCL | number of exclamation marks | GET THE CHIPMUNK ADVENTURE TODAY!!! |
| PUNC_QUES | number of question marks | I don't understand how the rating system works.?? |
| PUNC_DOT | number of dots | dragging us along for every aching second...",negative |
| SCORE_IN_REVIEW | score of the review | <br /><br />Story: 1/10 CG: 5/10 Acting:3/10<br>Whether it's Christmas or not, this movie gets a 500/10 |
| COUNT_REPEAT_CHAR | count of the repeated characters | E.g: hurrayyyyy, yayyyy |
| EMOTICON_POS | number of positive emoticons | XD =D =3 B^D c: C: Laughing |
| EMOTICON_NEG | number of negative emoticons | :E<br>Grimacing,nervous, awkward |

These features were carefully selected after referring to a research paper [3] that provides us with great insights on how a few emoticons are strong and reliable (and sometimes unique) signals of sentiment polarity and why one should take advantage of them to the sentiment analysis, the reference of which is put at the end of this document and the Exploratory data analysis helped us to gain insights about patterns

## Models

This project's main aim was to classify the review into positive or negative and find out which model performs better in predicting the outcome. As it is shown in table 1 of accuracies of each model, it was the result of the input data without pre-processing the dataset. It can be clearly seen that Logistic Regression performs well when compared to other models but not good enough.

After preprocessing the data and vectorizing the text in reviews, the resultant dataset had around 32,000 features. This large set of columns ensued the computational cost for all the models. Hence we reduced the number of samples to 10000 rows and was implemented to find the accuracies of each model. Though it consumed a lot of time to predict the outcome, the accuracies of each model were convincing (the accuracies can be seen in table 2, row name "After preprocessing").

Further, we tried checking the performance of all the models by excluding a few features in each run. The resultant accuracies can be seen in table 2 from row number 3 to 8 (the sign ~ indicates that particular feature was excluded during that execution). There was no improvement in the accuracy but these steps helped us realise the importance of each feature.

## Dimension Reduction

Furthermore, we reduced the number of features by applying PCA on the dataset. Principal Component Analysis (PCA) helps in reducing the dimension of features without losing any useful information from the features. Vectorizing the reviews resulted in a large number of columns and was shrunk to one Principal Component that had most of the information about the reviews by applying PCA. This Principal Component was then concatenated to other features such as “neg”, ”neu”, ”pos”, ”EMOTICON_POS”, ”EMOTICON_NEG” etc which was then fed into the models.

The models performed remarkably by giving phenomenal accuracies. (Refer table 2, row number 9).

## Conclusion

In the following section we have tabulated the accuracies of Support Vector Machine, KNN, Random Forest and Logistic Regression considering all the different features.

As we can see from the below table, improving training data features improved the accuracy across models. In our current iteration SVM has outperformed all the other algorithms.

| **No.** | **Features Selection** | **SVM** | **Naive Bayes** | **KNN** | **Random Forest** | **Logistic Regression** |
|---|---|---|---|---|---|---|
| 1 | Without Preprocessing | 0.864 | 0.866 | 0.685 | 0.852 | 0.884 |
| 2 | After data Pre-processing | 0.976 | - | 0.876 | 0.947 | 0.980 |
| 3 | neg | 0.967 | - | 0.825 | 0.922 | 0.971 |
| 4 | pos | 0.967 | - | 0.822 | 0.934 | 0.973 |
| 5 | Length stop words | 0.976 | - | 0.875 | 0.911 | 0.980 |
| 6 | PUNC_EXCL<br>PUNC_QUES<br>PUNC | 0.961 | - | 0.873 | 0.965 | 0.973 |
| 7 | SCORE_IN_REVIEW | 0.963 | - | 0.873 | 0.957 | 0.969 |
| 8 | EMOTICON_POS<br>EMOTICON_NEG | 0.962 | - | 0.874 | 0.948 | 0.968 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 9 | All Features after applying PCA | 0.98 | - | 0.91 | 0.977 | 0.975 |

## References


[1]. Lectures and materials provided in “CS 559A : Machine Learning”
[2]. Wikipedia: Sentiment Analysis https://en.wikipedia.org/wiki/Sentiment_analysis
[3]. Sentiment Expression via Emoticons on Social Media https://arxiv.org/pdf/1511.02556.pdf
[4]. Wikipedia: Emoticons https://en.wikipedia.org/wiki/List_of_emoticons