Final Report

For

**Term Project - Artificial Intelligence**

Sentiment Analysis of Movie Reviews

Prepared by:

Abdullah Bin Masood

December 30th, 2020

**Abstract:**

With the rise of the entertainment industry, a huge number of consumers give reviews, feedback and comments. Sentiment analysis plays an important role in decision making by subjective information extraction identifying positive and negative public opinions. Basically, it analyse people's thoughts and emotions based on linguistics. This is achieved by various classification algorithms. Sentiment analysis has vast applications including product reviews, movie review, customer feedback, surveys, social media influence and many more. In our project we concentrate on analysis of various movie reviews. Data is collected from top movie reviewing blogs (IMBD). It is filtered, tokenized and then passed through algorithms to give a precise analysis. Since the entertainment industry is a billion-dollar industry and online reviews, memes, tweets and comments are game changer for the profit and loss sentiment analysis is here to save the day. Through sentiment analysis, we can predict the intentions and emotions that whether they are in favour (positive) or against (negative) the product or in our case, a movie.

**Introduction:**

Sentiment analysis the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer’s attitude towards a particular topic, product, etc. is positive, negative, or neutral.

It involves extracting sentiments from text. Firstly, the reviews are tokenized into words. Irrelevant words are filtered out which may include “a”, “the” etc.

We have used the stop words list from NLTK. We proposed to do sentiment analysis using BOW model using Raw counts and TF-IDF vectorizers. However, we were successful to implement the extension to our project as mentioned in the project proposal i.e. TextBlob and Neural Network technique. NB model for computational purposes which is based on Bayes theorem of probability.

* This algorithm works quickly and can save a lot of time.
* Naive Bayes is suitable for solving multi-class prediction problems.
* If its assumption of the independence of features holds true, it can perform better than other models and requires much less training data.
* Naive Bayes is better suited for categorical input variables than numerical variables.

All the above-mentioned advantages suggest that naive bayes theorem is one of the best models for text classifications which we are using in our project. Moreover, we also used the Bag of Words (BOW) TF-IDF model . A vector of vocabulary is established from sentences. Every time a word from vocabulary repeats itself it simply increments the location of that word in the matrix.

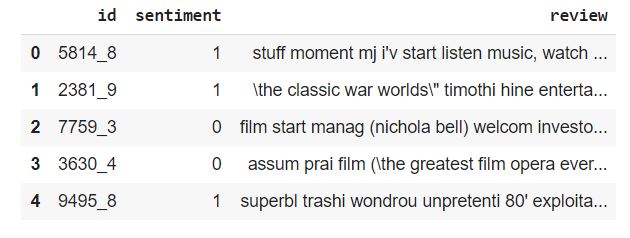
CNN and Textblob were also implemented in order to capture the sentiments and finally compare them to find which of the following is comparatively more accurate.

The sentiment function of textblob returns two properties, polarity, and subjectivity. Polarity is a float which lies in the range of [-1,1] where 1 means positive statement and -1 means a negative statement. In our case, we used threshold of **0** to divide the polarity. Subjective sentences generally refer to personal opinion, emotion or judgment whereas objective refers to factual information. Subjectivity is also a float which lies in the range of [0,1].

At the end, we also implemented Neural Network to run it on same data set and hence we were able to compare the results.

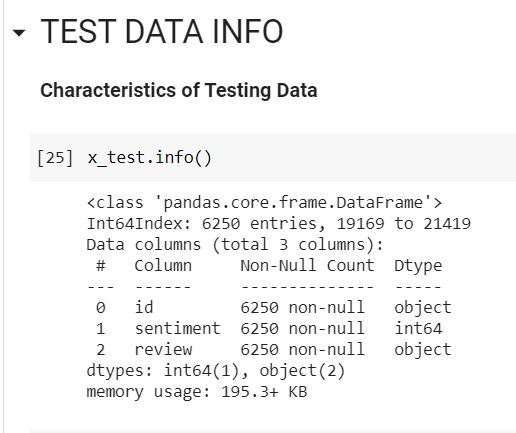
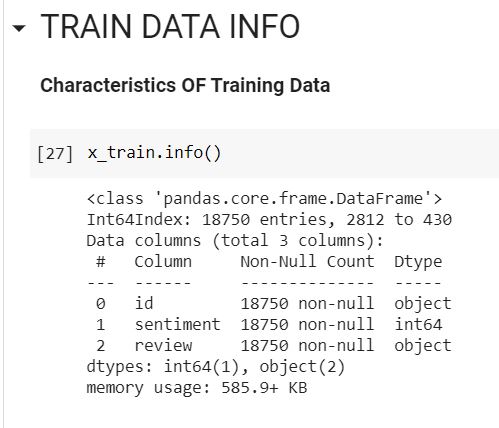
**Dataset:**

The dataset was collected by IMDB which considered to be one of the top movie rating/review websites. It is a collection of 25,000 positive and negative reviews. The project lies under the scope of Supervised Learning since the dataset has labels of 1 and 0 where 1 represents a positive review and 0 represents negative review.

* First column shows review number
* Second column is for review ID
* Third is for review type (AKA sentiment)
* Fourth column is for the actual comment body of the review.

We Splited the data into train and test set by using **“train\_test\_split(DataSet)”** of scikit. By default it will split into 75% training and 25% test data.

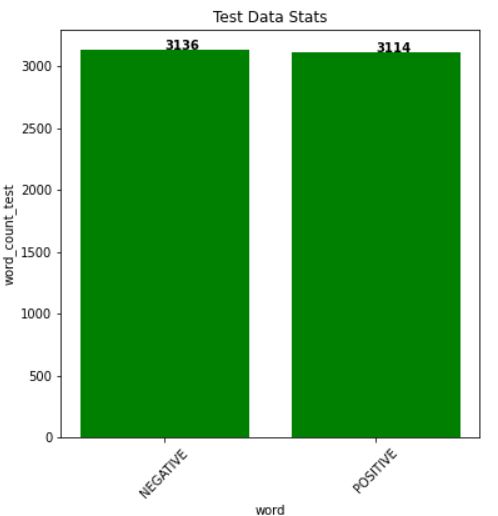
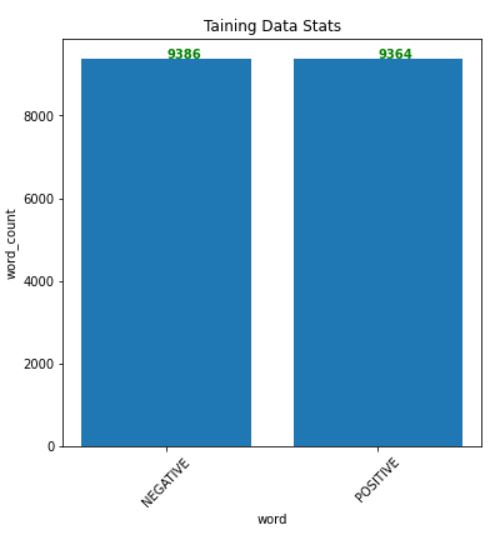
The characteristics of this split is as follow:

****

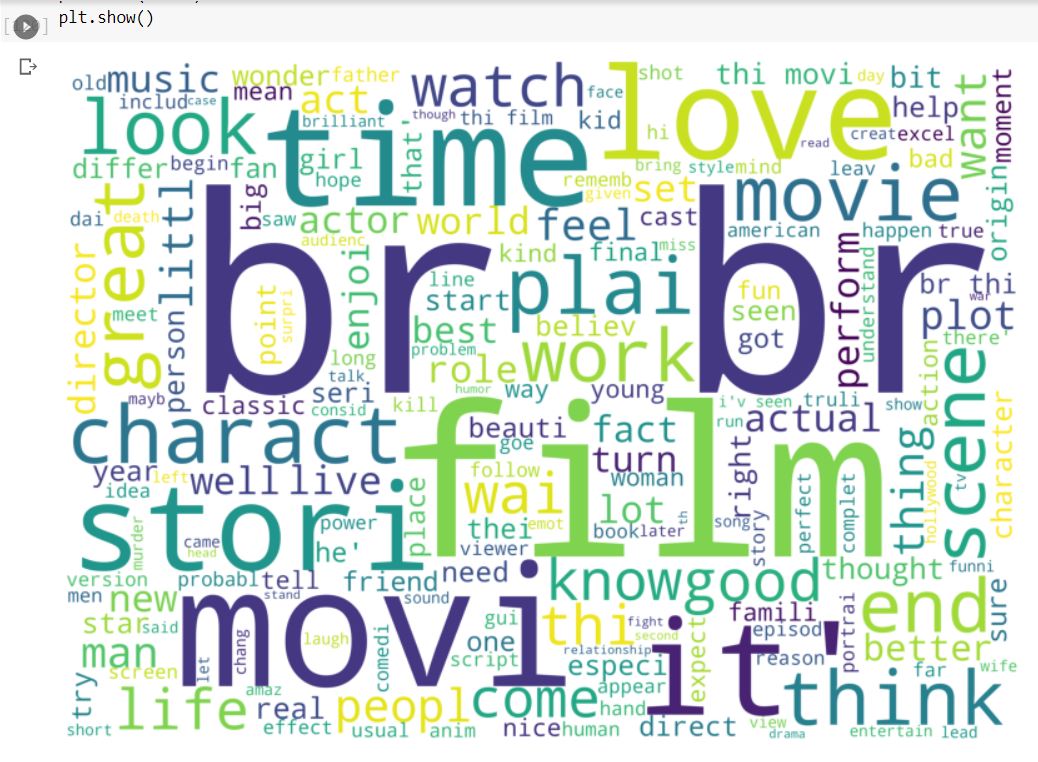
Then we analyse the stats of positive and negative reviews in both datasets.

Negative 9386 Positive 3136

Positive 9364 Negative 3114

dtype: int64 dtype: int64

Wordcloud representation of our dataset



**Methodology**

**Bag Of Words Model:**

We can define the bag-of-words model as a simplifying representation used in natural language processing and information retrieval. It is also known as the Vector Space Model. In this model, a text such as a sentence or a document is represented as the bag of its words by disregarding grammatical errors and even word order. We maintain the multiplicity in case of word order. For example, consider two sentences.

Sentence 1: The cat is moving towards the dog.

Sentence 2: The dog is eating food.

The list of words that are formed from the above two

Sentences will be:

{Cat, moving, towards, dog, eating, food}

This list in a bag is unsorted and that forms a “Bag of words”.

**Naïve Bayes:**

Abstractly, Naïve Bayes’ is a conditional probability model. Despite its simplicity and strong assumptions, the naïve Bayes’ classifier has been proven to work satisfactorily in many domains. Bayesian classification provides practical learning algorithms and prior knowledge and observed data can be combined. In Naïve Bayes’ technique, the basic idea to find the probabilities of categories given a text document by using the joint probabilities of words and categories. It is based on the assumption of word independence. The starting point is the Bayes’ theorem for conditional probability, stating that, for a given data point x and class C: P (C / x) = P(x/C)/P(x).

**Bag of words based on TfIDF:**

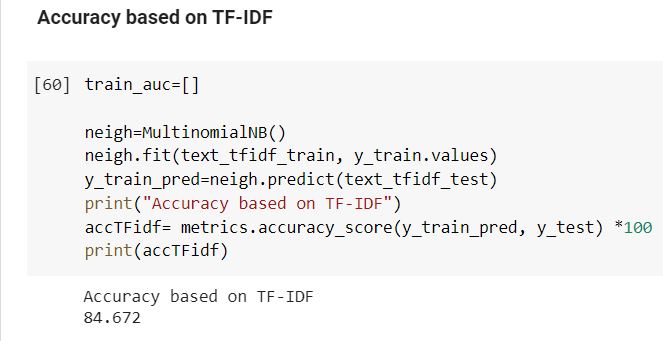
The **inverse document frequency (IDF)**of the word across a set of documents. This suggests how common or rare a word is in the entire document set. The closer it is to 0, the more common is the word. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.

So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

Multiplying these two numbers results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document.

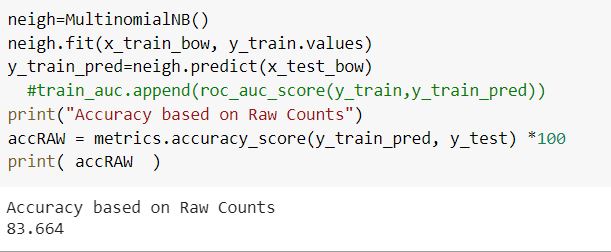
To put it in mathematical terms, the TF-IDF score is calculated as follows:

**IDF=1+log(N/dN)**



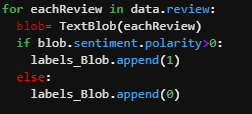
**Bag of words based on raw counts:**

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work. Parameters alphafloat, default=1.0. **Multinomial Naïve Bayes** uses term frequency i.e. the number of times a given term appears in a document. ... After normalization, term frequency can be used to compute maximum likelihood estimates based on the training data to estimate the conditional probability.



**TextBLOB:**

The sentiment function of textblob returns two properties, polarity, and subjectivity. Polarity is a float which lies in the range of [-1,1] where 1 means positive statement and -1 means a negative statement. Subjective sentences generally refer to personal opinion, emotion or judgment whereas objective refers to factual information. Subjectivity is also a float which lies in the range of [0,1]



A review is classified as positive if its sentiment and polarity are both greater than 0. Here **0** is set **as threshold value.** Otherwise it's considered negative. We achieved the following results after applying this classification technique on a filtered dataset of 25,000 objects with an accuracy of 68%.

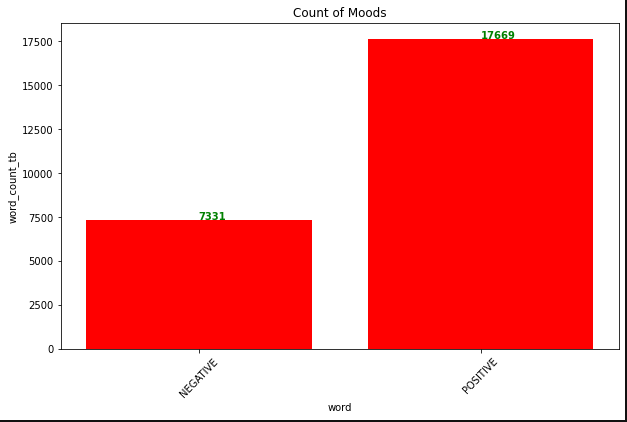
Here, using textblob library we can analyse the stats of positive and negative reviews classified by this method.

Which are as follow:

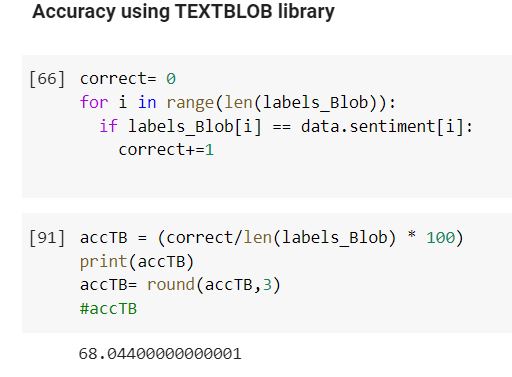
Negative 7437

Positive 17563

dtype: int64



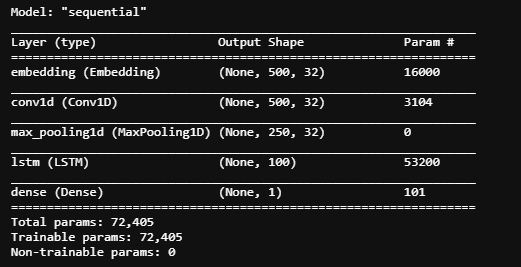
Using the textblob library, the accuracy we got is:



The accuracy we got through this built in library is quite low since it is solely dependent on threshold value set, here we set it as 0. And on the basis of its comparison with the dataset we calculate the accuracy. The accuracy can vary if change the threshold value.

**Neural Network:**

Neural Network is a classic AI algo used for almost the majority of problems in ML. We used a 5 layered model to classify the data into 2 classes (positive and negative reviews). Each layer is named in the figure below.



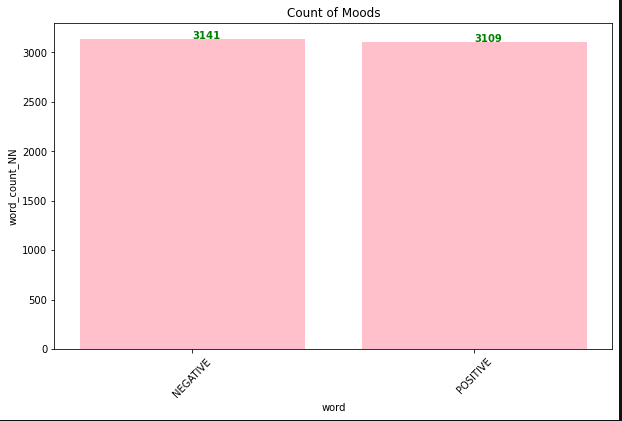
Here, using NN we can analyse the stats of positive and negative reviews classified by this method.

Which are as follow:

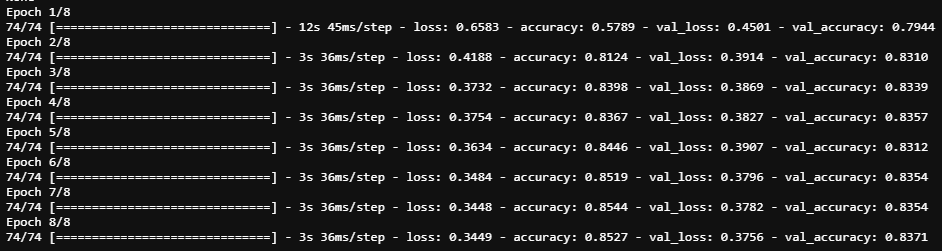
Positive 3335

Negative 2915

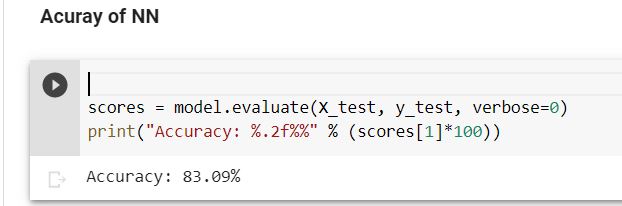
dtype: int64



While training our model epochs were set to 8 and steps per epochs are 74.



We managed to achieve an accuracy of 83% which produced the following results.



**Analysis:**

Since the purpose of our project was to achieve a classification model with a reasonable accuracy, we successfully optimized our model to achieve an accuracy of ≈ 84% which is fair enough for Naïve Bayes model and Bag-of-Words.





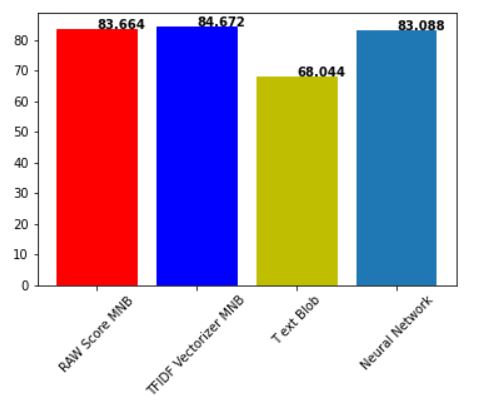
It learns word portrayals catching semantic and opinion data. The Bag of Words display takes in a vocabulary from the majority of the archives, at that point models each report by tallying the circumstances each word shows up.

For CNN our model was able to classify data with the following accuracy.



As for textblob method





The graph is the portrayal of accuracy comparison of all the implemented algorithms. (accuracy on y and algos on x-axis).

**Conclusion:**

Out of all naive Bayes had an optimal accuracy since it can train on small amounts of data with high accuracy in multi-class prediction problems. Moreover it's faster in performance. Raw counts and TFIDF score here are giving almost same results in this case since we have single dataset document. However, TFIDF score better when we have multiple documents to do analysis. But still it was successful in giving best accuracy. Using TextBlob library, the results were quite low since it is builtin library, we can improve the results by improving the data cleaning process. More the clean sentence is passed, more accurately it will classify it as positive or negative review, since a review can contain a lot of unimportant words and score is calculated on the bases of a whole sentences. Where on the other hand, TF-IDF gives the best accuracy result since it focus on the importance if word that how much a word is common or rare, that is why it is able to give best results by focusing on Important words in the review. CNN requires comparatively more time with a slight increase in accuracy or we can change the ratio of training data for more accurate results. Changing the number of epochs can also increase our accuracy.