|  |
| --- |
| Worcester Polytechnic Institute |
| CASE STUDY 3 |
| TEAM 03- Karan, Deepan, Dhaval, Bhakti, Rohitpal  Image result for TEXTUAL ANALYSIS OF MOVIE REVIEW |

|  |
| --- |
| DS 501  11-17-2016 |

Introduction:

The goal of this case study is to analyze the movie review textual data set, make conjectures, support or refute those conjectures with data. To achieve this goal, we created a word dictionary where every word will have a weight and accordingly that will determine the polarity, (+ve or -ve) of the sentence. Weight of a word is determined by its number of occurrence in that document and the other comparing document.

We split our dataset into 2 sets. Training set, which is fed into the system so that the model can learn from it and Test Set – Based on learnings from training set, we will predict the outcome of the test set. We use some comments as training set, determine the accuracy of our model by analyzing the result of the test set. We will find the most accurate parameters, and based on that we will evaluate the performance of the test set.

Why this topic is interesting or important to you?

This case study is based on text analysis. Text mining, also referred to as text data mining, equivalent to text analytics, refers to the process of deriving high-quality information from text. High-quality information is typically derived through the devising of patterns and trends through means such as statistical pattern learning. (Source: Wikipedia)

Unstructured data is a large part of big data and we team members as a bunch of budding Data Scientists are interested in Unstructured Data. Unstructured data are things like text (say, a survey or tweets), or video or voice recording. The generally accepted saying is that structured data represents only 20% of the information available to an organization. That means that 80% of all the data is in unstructured form. If businesses are gaining value from analyzing only 20% of their data, then there is a massive potential waiting to be leveraged in the analysis of unstructured data.

Major companies have made clear moves showing the importance of text analytics.   For example, IBM has been pushing their Watson platform, and recently acquired AlchemyAPI to augment the analytics side of Watson. In another example, Microsoft purchased Equivio, a text analytics company focusing on eDiscovery. Businesses use data and text mining to analyze customer and competitor data to improve competitiveness; the pharmaceutical industry mines patents and research articles to improve drug discovery

This case study is basically built on analyzing data to discover interesting patterns, extract useful knowledge, and support decision making, with an emphasis on statistical approaches that can be generally applied to random text data in with minimum human effort.

Text mining is most beneficial when:

1) Summarizing documents- Creating Wordle

2) Extracting concepts from text- Social Media, Understanding customers

3) Indexing text for use in predictive analytics – Search Engine

What data did you collect? How did you analyze the data?

**Q1:**

We used fetch\_data.py file to get the data. We used load files function from sci-kit package using that we loaded the files.

We used random split and set 25% of the data set as test set and 75% as training set.

We tried 10-folds cross validation to lessen the biasness of that dataset so that the dataset can be efficiently used as test set and training set. Unfortunately, we couldn’t use this approach as it took a long time to create this model and implement it.

Training set has 1500 reviews and Test set has 500 reviews

We encountered many stop words in the reviews. Stop Words are words which do not contain important significance to be used in Search Queries. Usually these words are filtered out from search queries because they return vast amount of unnecessary information. To remove these stop words (the, a, an, etc.) we are using count vector to get the matrix of token counts and remove the stop words.

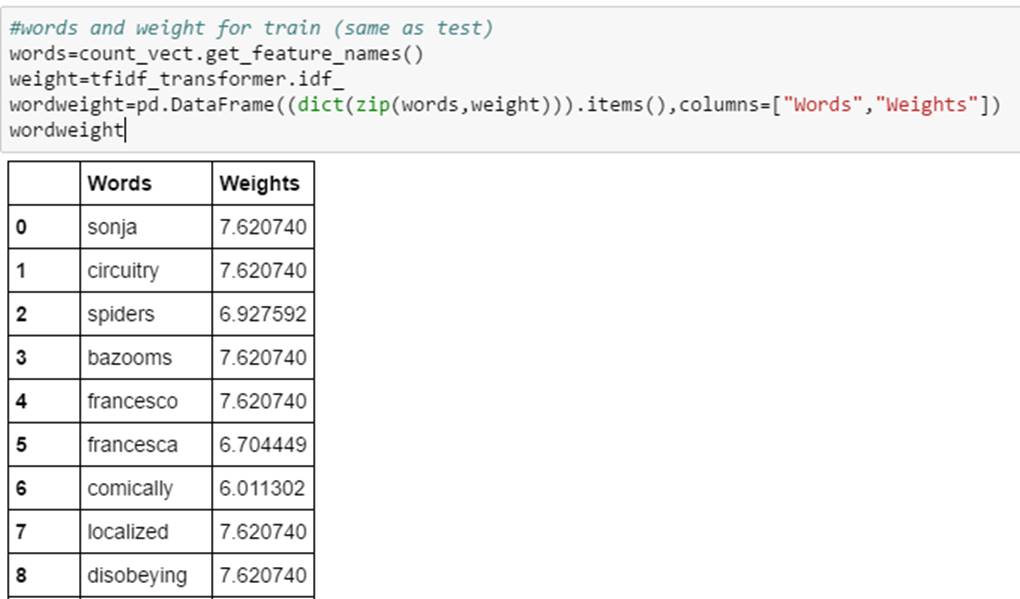
We first thought of using Count Vectorizer. Count Victorizer helps in text processing, tokenizing and filtering of stop words.

But, we found that using count as a measure has a disadvantage, that is, the longer documents will have higher average count values than shorter documents on the same topic, which will decrease accuracy of the model and introduce unwanted bias in the model.

To avoid this, we choose, term frequency times inverse document frequency (tf-idf)

This function, divides the number of occurrences of each word in the document by the total number of words in that document. This helps in getting better accuracy by omitting unwanted biases.

We fit our training data using tf-idf transformer function, which calculates the count matrix, which is dictionary of words with weights corresponding to each word. We have also displayed the dictionary to analyze the result in more detail.



Training the Classifier:

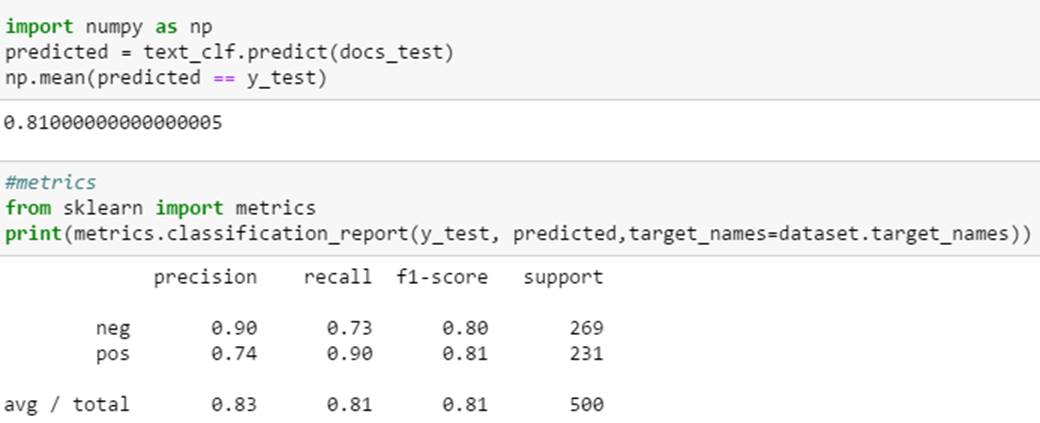
We used Naïve Bayes Classifier, where we gave our training set and training result as input. Once, the Naïve Bayes classifier created a model on the training set, we predicted the results of the test data.

To predict these results we used tf-idf transformer as our dictionary.

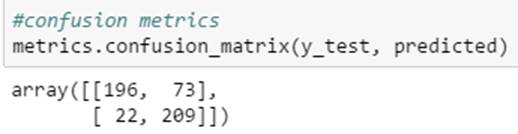
So all this process, of creating count vectorizer, tf-idf transformer and classifier, can be combined using Pipeline function of Sci-Kit learn.

Pipeline helps in ease the process by minimizing the code length. Below is the screenshot of the result of our model:

Here we can see that 81% of the data was correctly classified.



Confusion Matrix:



Above mentioned was one of the methods to segregate the Negative and Positive reviews using tf-idf. This can also be done using the N-gram Range function

Great search: Enables us to test various parameters simultaneously and gives the best fit result as the output.

**Q2:**

Define TF:It divides each word in a document by the total number of words in that document.

We further enhanced TF values by reducing the weight for the words that occur in many documents, this is required, the words with highest occurrences in all the documents are mostly less informative (Eg: the, is, of etc.) as compared to the words that have less occurrence in the documents. This reducing of weighs is called Term frequency time’s inverse document frequency (tf-idf).

tf-idf Vectorizer Class:

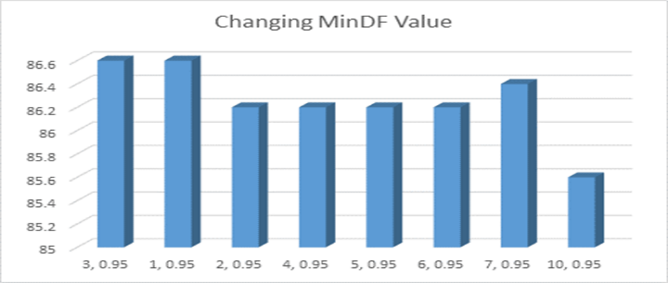
This class does the same work as the tf-idf and TF Vectorizer combined. It gives the matrix of tf-idf as the output when fed with Training set as the input. This is a class and the input parameters for this class are max\_df, min\_df and n-gram range.

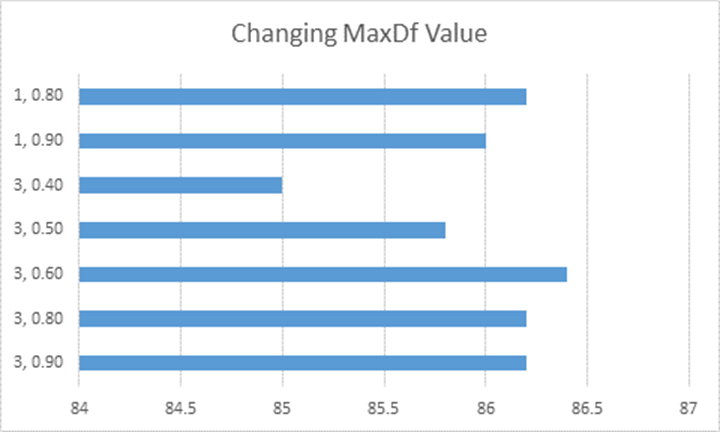
df refers to document frequency, that is, the percentage of documents that contain the term.

max\_df**:** It ignores the words that have a occurrence frequency less than the min\_df value. max\_df is used for removing terms that appear too frequently, also known as "corpus-specific stop words"

min\_df: It can be either 0, 1, 2 (int) or the range between 0 and 1. It ignores the words that have occurrence frequency greater than the max\_df value. min\_df is used for removing terms that appear too infrequently.

As we increase the value of min\_df and decrease the value of max\_df , the number of tokens in a matrix decreases, that is, the number of words in the dictionary decreases. This helps us to set values depending on the domain knowledge of the document and efficiently reducing the number of words in the dictionary for obtaining better observations and analysis.

 As observed in the graph, for most of the higher values of Min DF (Keeping Max Df constant) we get a decreasing trend in the accuracy (Mean of the model). Hence we take 1 as the value of Min DF for our further results.

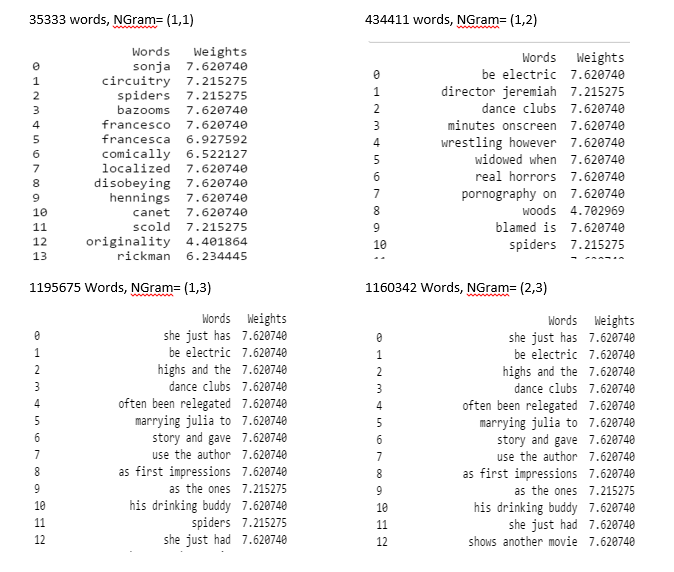


Similarly, we get 1,0.9 and 3,0.9 as the best value which results in highest accuracy.

n-gram range:

n-gram range returns the set of occurrence of words depending on the value of n. When N=1, this is referred to as unigrams and this is essentially the individual words in a sentence. When N=2, this is called bigrams and when N=3 this is called trigrams. It can be (Min value, Max Value). It considers both the Min value and the Max value. It returns both the values from the data dictionary.

As we increase the Max value the n-gram, the number of words in the matrix increase, keeping the Min value constant. Keeping the Max value constant, when the Min value is increase the number of words in the matrix increases, that is only when Min value <= Max value.

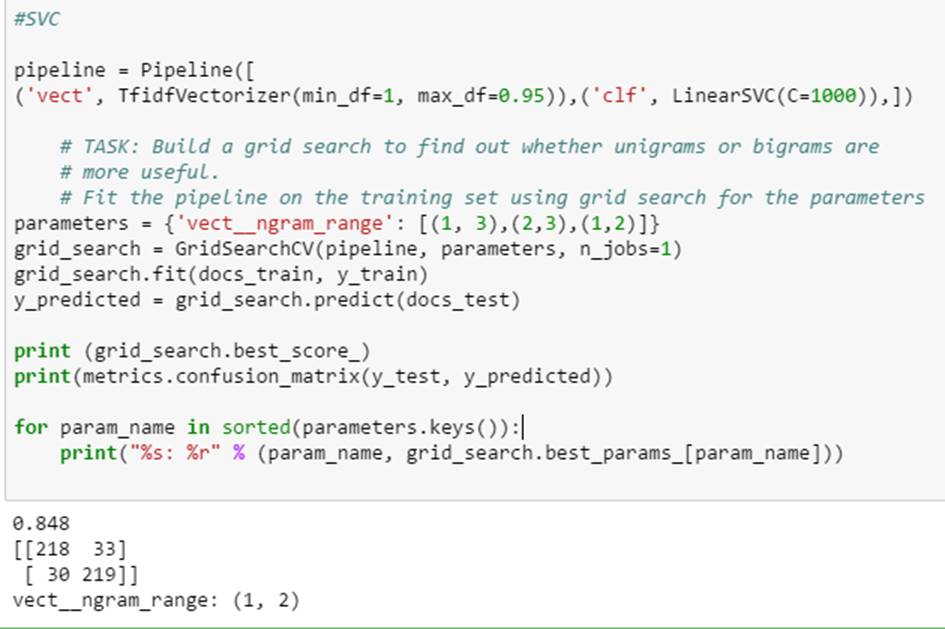


**Q3:**

Based on the analysis on n-gram and tf-idf value, we decided our tf-idf value to be (1, 0.95) and n-gram value to be (1, 2), (1, 3), (2, 3). We choose these values as they gave us the maximum accuracy for the model. Along with these parameters we will use tf-idf vectorizer class to fit and transform the training sets in to the word dictionary, which will help to create a better model. Also, we have used tf-idf transform function on the test data. We now use 2 classifiers:

1. Linear SVC:

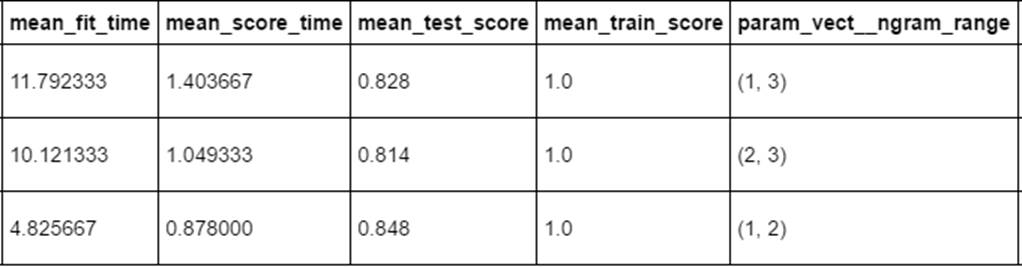
The result of Linear SVC is as follows:



As we can observe, we got the best result for n gram (1, 2). 84.8% classes were correctly classified.

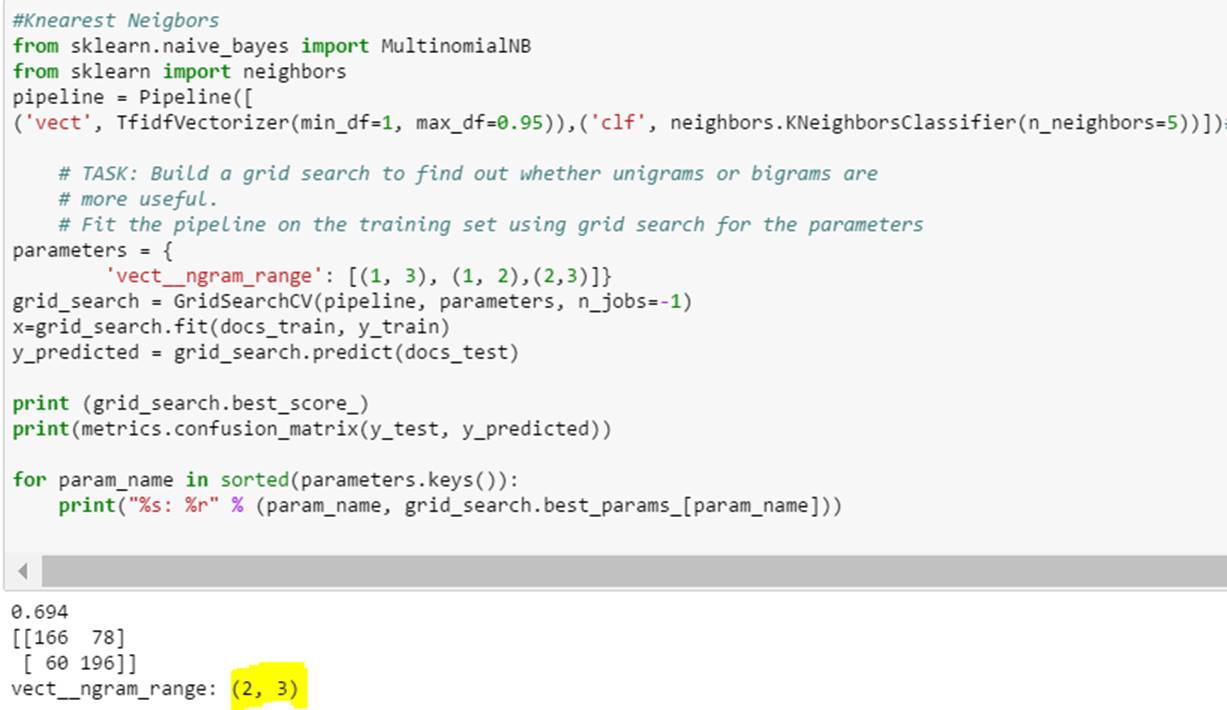
(1, 3) and (2, 3) didn’t give the best results as (2, 3) doesn’t contain (important) single words in the dictionary, which affects the accuracy of the model and (1, 3) contains insignificant 3 words, which leads to decrease the accuracy.

As we can see, the **mean\_test\_score** is the highest and **mean\_fit\_time** is the lowest for (1, 2), hence, we get best results for this class.

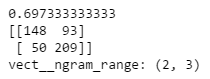


1. K- Neighbors :

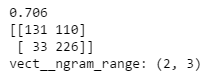
As we can see, 69.4 % of classes were correctly classified with n=5 using Knearest-neighbors.



This is the result with K=9 for K nearest Neigbours

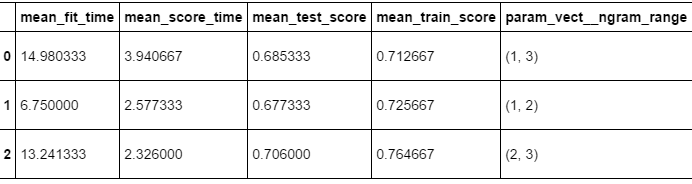


N=19

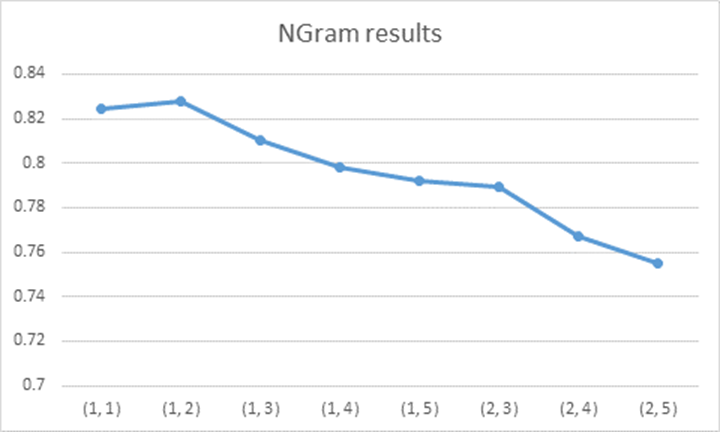


There is a slight increase in the efficiency but it’s not considerable because as we increase the value of K, the time required to process also increases. Also, by trying different value of K, we can conclude that as we decrease the value of K, the efficiency of model decreases. Moreover, for this model, combination of 2 and 3-word dictionary seems to be more appropriate than others.

Below are the other NGram results:



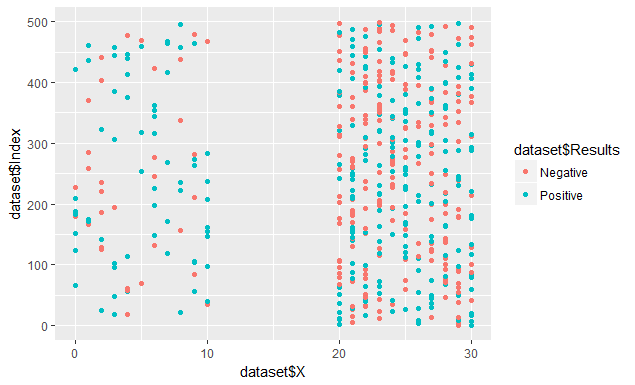
**mean\_test\_score** for NGram (2,3) is the highest compared to others.

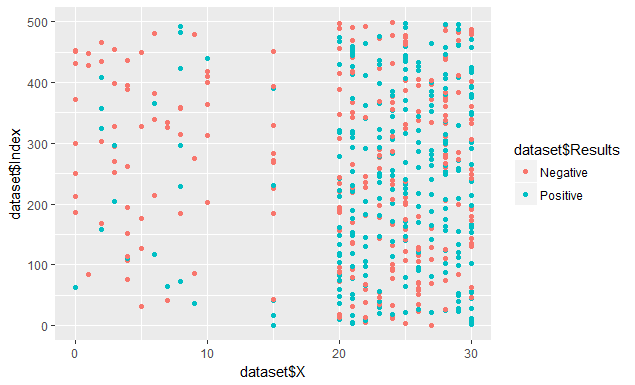
**These are the results for different value of NGram. We get the best value for 1,2 as shown in the figure**

**Q4:**

After trying to find correlation between various features like Length of Review, Number of Words in a review and word count, we concluded that dictionary is must to distinguish negative and positive words. Previous methods were tried just to see how bad can be the results.

We first distinguished the Dictionary of words or bag of words to Positive or Negative. For this, we chose the approach which every data scientist has to follow. First, we tried to learn from the training set to distinguish negative and positive words. This we did by tracing each word count in each review and summing all the counts. After this we tried using Mean of the positive and negative words counts to distinguish. But we weren’t able to classify them with acceptable level of accuracy. Then we considered count of those words which are not in both positive and negative dictionaries. Below is the result:



Over here we can see more positive on left side and more negative on right side. To achieve more accuracy, rather than considering the word count, we tried to use word frequency of positive and negative words in a particular review. Below are the results:

This results do decent work in separating two types of reviews. Also, it has some comments which cannot be identified as either negative or positive.

Further, we tried using tf-idf concept and tried to finding out weights of each word. We divided each word by the total number of words in the document. Now, we add the weights instead of count and we get the following plot:

