**PROCESSING AND CLASSIFICATION OF THE KAGGLE MOVIE REVIEW DATA**

**IST 664 FINAL PROJECT**

**STEPHEN OMONDI**

**June 15, 2020**

**Course Instructor: Professor Paloma**

**INTRODUCTION**

The Kaggle Movie Review Data was taken from the original Pang and Lee movie review corpus based on reviews from the Rotten Tomatoes web site. Socher’s group used crowd-sourcing to manually annotate all the subphrases of sentences with a sentiment label that went into this movie review library.

**The sentiment labels used were:**

0 - negative

1 - slightly negative

2 - neutral

3 - slightly positive

4 – positive

This analysis uses the training data “train.tsv”, and test data “test.tsv”. The training data file has 156,060 phrases, and the early part of the analysis chooses an appropriate subset for processing and training.

**Objective**

The goal of this analysis is to predict the sentiments of each review using algorithmic approach and make a comparison of results by tweaking underlying parameters such as the effect on the classification based on different filtering and preprocessing approaches. Finally, a summary comparison of different algorithmic classifiers, such as Naïve Bayes classification, Random Forests, Decision Trees and more from the Sci-Kit Learn ecosystem are compared along common measures of prediction accuracy to determine the winner.

**STEP 1: FETCH AND POSITION REVIEWS FOR ANALYSIS**

**- Fetching data from train.tsv**

First, begin by reading the training tab separated values file. The file has 156,060 phrases. 3000 random phrases were selected to avoid overlaps and to display a wider segment of the total phrases some of which are shown below, along with their sentiments.

[['jettisoned some crucial drama', '2'],

[', mournfully brittle delivery', '0'],

["about 3\\/4th the fun of its spry 2001 predecessor -- but it 's a rushed , slapdash , sequel-for-the-sake - of-a-sequel with less than half the plot and ingenuity",

'1'],

['buried somewhere inside its fabric , but never clearly seen', '1'],

['for gas', '2'],

['see another car chase , explosion or gunfight again', '2'],

['a Hallmark commercial', '2'],

["... gives his best screen performance with an oddly winning portrayal of one of life 's ultimate losers .",

'3'],

['typical American horror film', '2'],

['true to his principles', '3']]

**- Considerations for the appropriate tokenization approach**

Three tokenization approaches are explored below in order to set the stage for forthcoming experiments:

* **NLTK Word Tokenize**

This approach preserves abbreviated words like U.S.A but splits negative words like *can’t* into two parts, *can* and *‘t.*

* **NLTK Wordpunct tokenizer**

This tokenizer treats punctuations as words on their own. For example, words like ‘U.S.A.’ becomes six words- ‘U’,’.’,’S’,’.’,’A’,’.’ and ‘ca n’t’ becomes four words- ‘ca’,’n’, ’’’ ,’t’.

* **Sklearn Countvectorize tokenizer**

This tokenizer removes all single character words. For example, the word *won’t* became *won* and all abbreviated words were removed, e.g. U.K was omitted altogether. This approach may not be ideal when handling negative sentiments or negative words as they are lost in the tokenization which may lead to a different meaning of the word.

*NLTK Word Tokenize* appears to preserve the original word structure the most and is going to be used in further analyses.

**STEP 2: PRE-PROCESSING REVIEW TEXT FOR ANALYSIS**

**- Convert to Lowercase**

Convert all reviews to lowercase. This is important because most of the algorithms used within NLTK are case sensitive.

[('jettisoned some crucial drama', '2'),

(', mournfully brittle delivery', '0'),

("about 3\\/4th the fun of its spry 2001 predecessor -- but it 's a rushed , slapdash , sequel-for-the-sake - of-a-sequel with less than half the plot and ingenuity",

'1'),

('buried somewhere inside its fabric , but never clearly seen', '1'),

('for gas', '2'),

('see another car chase , explosion or gunfight again', '2'),

('a hallmark commercial', '2'),

("... gives his best screen performance with an oddly winning portrayal of one of life 's ultimate losers .",

'3'),

('typical american horror film', '2'),

('true to his principles', '3')]

**- Clean the reviews to include negative / contraction words**

Expand the stop words list by removing contraction words like *can't* to *cannot*. In effect, this process extends the stop words with apostrophes. This is important as it helps the NLTK Word Tokenizer perform better.

[('jettisoned some crucial drama', '2'),

(' mournfully brittle delivery', '0'),

('about \\th the fun of its spry predecessor but it is a rushed slapdash sequelforthesake ofasequel with less than half the plot and ingenuity',

'1'),

('buried somewhere inside its fabric but never clearly seen', '1'),

('for gas', '2'),

('see another car chase explosion or gunfight again', '2'),

('a hallmark commercial', '2'),

(' gives his best screen performance with an oddly winning portrayal of one of life s ultimate losers ',

'3'),

('typical american horror film', '2'),

('true to his principles', '3')]

**- Removing Punctuations, Numbers and Special characters**

This is necessary since punctuations and numbers will not be necessary for sentiment analysis.

[('jettisoned some crucial drama', '2'),

(' mournfully brittle delivery', '0'),

('about \\th the fun of its spry predecessor but it is a rushed slapdash sequelforthesake ofasequel with less than half the plot and ingenuity',

'1'),

('buried somewhere inside its fabric but never clearly seen', '1'),

('for gas', '2'),

('see another car chase explosion or gunfight again', '2'),

('a hallmark commercial', '2'),

(' gives his best screen performance with an oddly winning portrayal of one of life s ultimate losers ',

'3'),

('typical american horror film', '2'),

('true to his principles', '3')]

**- Removing stopwords (updating the stopwords list**

**The standard NLTK English stop words is used in this analysis augmented by additional stop words not available in the list. These additional stop words are:**

**['not', 'no', 'can','has','have','had','must','shan','do', 'should','was','were','won','are','cannot','does','ain', 'could', 'did', 'is', 'might', 'need', 'would']**

**The clean reviews after removing all stop words looks like this:**

[('jettisoned some crucial drama', '2'),

(' mournfully brittle delivery', '0'),

('about \\th the fun of its spry predecessor but it is a rushed slapdash sequelforthesake ofasequel with less than half the plot and ingenuity',

'1'),

('buried somewhere inside its fabric but never clearly seen', '1'),

('for gas', '2'),

('see another car chase explosion or gunfight again', '2'),

('a hallmark commercial', '2'),

(' gives his best screen performance with an oddly winning portrayal of one of life s ultimate losers ',

'3'),

('typical american horror film', '2'),

('true to his principles', '3')]

**- Tokenization using NLTK Tokenizer**

**Applying NLTK Word Tokenization before and after processing yields different results as shown below. Preprocessing significantly reduces the token vocabulary length. In this sample the tokens count moved from *755* before preprocessing to *702* after preprocessing.** Only first 70 tokens are shown and will be used in subsequent analyses to better manage existing memory.

|  |  |
| --- | --- |
| **Without pre-processing** | **With pre-processing** |
| ['jettisoned', 'some', 'crucial', 'drama', ',', 'mournfully', 'brittle', 'delivery', 'about', '3\\/4th', 'the', 'fun', 'of', 'its', 'spry', '2001', 'predecessor', '--', 'but', 'it', "'s", 'a', 'rushed', ',', 'slapdash', ',', 'sequel-for-the-sake', '-', 'of-a-sequel', 'with', 'less', 'than', 'half', 'the', 'plot', 'and', 'ingenuity', 'buried', 'somewhere', 'inside', 'its', 'fabric', ',', 'but', 'never', 'clearly', 'seen', 'for', 'gas', 'see', 'another', 'car', 'chase', ',', 'explosion', 'or', 'gunfight', 'again', 'a', 'Hallmark', 'commercial', '...', 'gives', 'his', 'best', 'screen', 'performance', 'with', 'an', 'oddly'] | ['jettisoned', 'some', 'crucial', 'drama', 'mournfully', 'brittle', 'delivery', 'about', '\\th', 'the', 'fun', 'of', 'its', 'spry', 'predecessor', 'but', 'it', 'is', 'a', 'rushed', 'slapdash', 'sequelforthesake', 'ofasequel', 'with', 'less', 'than', 'half', 'the', 'plot', 'and', 'ingenuity', 'buried', 'somewhere', 'inside', 'its', 'fabric', 'but', 'never', 'clearly', 'seen', 'for', 'gas', 'see', 'another', 'car', 'chase', 'explosion', 'or', 'gunfight', 'again', 'a', 'hallmark', 'commercial', 'gives', 'his', 'best', 'screen', 'performance', 'with', 'an', 'oddly', 'winning', 'portrayal', 'of', 'one', 'of', 'life', 's', 'ultimate', 'losers'] |

**- Stemming and Lemmatization**

***Stemming* is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma.**

***Lemmatization* involves the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.**

**The table below shows the effect** **of both stemming and lemmatization on the tokens thus far:**

|  |  |
| --- | --- |
| **Lemmatization** | **Stemming** |
| jettisoned  some  crucial  drama  mournfully  brittle  delivery  about  \th  the  fun  of  it  spry  predecessor  but  it  is  a  rushed  slapdash  sequelforthesake  ofasequel  with  le  than  half  the  plot  and  ingenuity | jettison  some  crucial  drama  mourn  brittl  deliveri  about  \th  the  fun  of  it  spri  predecessor  but  it  is  a  rush  slapdash  sequelforthesak  ofasequel  with  less  than  half  the  plot  and  ingenu |

**STEP 3: DEVELOPING FEATURES IN THE NOTATION OF NLTK**

**- Bag of Words Feature**

**This is a collection of words in the corpus where only a few of the most frequent words are considered for feature engineering. Unigram, Bigram and Negation features are built and explored further in the experiments section.**

**- Building Unigram Word Features**

|  |  |
| --- | --- |
| **Top 10 Unprocessed tokens word features** | **Top 10 Pre-Processed tokens word features:** |
| ['the', ',', 'a', 'of', 'and', "'s", 'to', '.', 'in', 'it'] | ['the', 'a', 'of', 'and', 'to', 'in', 'it', 'is', 's', 'for'] |

**- Building Bigram Word Features**

|  |  |
| --- | --- |
| **Top Unprocessed tokens word features (BIGRAMS)** | **Top Pre-processed tokens word features (BIGRAMS)** |
| [('as', 'as'),  ('one', 'of'),  ('it', "'s"),  ('of', 'the'),  ("'s", 'a'),  ('the', 'of'),  ('to', 'the'),  ('the', ',')] | [('it', 'is'),  ('as', 'as'),  ('one', 'of'),  ('of', 'the'),  ('is', 'a'),  ('to', 'the'),  ('the', 'of')] |

**- Unigram Features (Baseline feature for future comparison):**

**The Unigram Features is a dictionary where each element is a word (obtained from bag of words defined earlier) with a Boolean value indicating whether that word occurred in document or not. The feature label will be ‘has(keyword)’ for each keyword (aka word) in the bag of words set.**

**An Example, looking at the top 20 most common word features from the unprocessed tokens**

{'has(the)': True,

'has(,)': True,

'has(a)': True,

'has(of)': True,

'has(and)': True,

"has('s)": True,

'has(to)': True,

'has(.)': True,

'has(in)': True,

'has(it)': True,

'has(for)': True,

'has(one)': True,

'has(that)': True,

'has(as)': True,

'has(about)': True,

'has(but)': True,

'has(film)': True,

'has(at)': True,

'has(like)': True,

'has(is)': True}

**- Bigram Features:**

**This is a feature dictionary of bigrams. Example Bigram sets without preprocessing are:**

{'has(the)': True,

'has(,)': True,

'has(a)': True,

'has(of)': True,

'has(and)': True,

"has('s)": True,

'has(to)': True,

'has(.)': True,

'has(in)': True,

'has(it)': True,

'has(for)': True,

'has(one)': True,

'has(that)': True,

'has(as)': True,

'has(about)': True,

'has(but)': True,

'has(film)': True,

'has(at)': True,

'has(like)': True,

'has(is)': True,

'bigram(as as)': False,

'bigram(one of)': False,

"bigram(it 's)": False,

'bigram(of the)': False,

"bigram('s a)": False,

'bigram(the of)': False,

'bigram(to the)': False,

'bigram(the ,)': False}

**Building Feature Sets – a summary:**

**Below is an exploration of feature sets before and after processing with Unigrams and Bigrams.**

**Unigramsets\_without\_preprocessing -**

({'has(the)': False, 'has(,)': False, 'has(a)': True, 'has(of)': False, 'has(and)': False, "has('s)": False, 'has(to)': False, 'has(.)': False, 'has(in)': False, 'has(it)': False, 'has(for)': False, 'has(one)': False, 'has(that)': False, 'has(as)': False, 'has(about)': False, 'has(but)': False, 'has(film)': False, 'has(at)': False, 'has(like)': False, 'has(is)': False}, '2')

**Unigramsets\_with\_preprocessing -**

({'has(the)': False, 'has(,)': False, 'has(a)': True, 'has(of)': False, 'has(and)': False, "has('s)": False, 'has(to)': False, 'has(.)': False, 'has(in)': False, 'has(it)': False, 'has(for)': False, 'has(one)': False, 'has(that)': False, 'has(as)': False, 'has(about)': False, 'has(but)': False, 'has(film)': False, 'has(at)': False, 'has(like)': False, 'has(is)': False}, '2')

**Bigramsets\_without\_preprocessing -**

({'has(the)': False, 'has(,)': False, 'has(a)': True, 'has(of)': False, 'has(and)': False, "has('s)": False, 'has(to)': False, 'has(.)': False, 'has(in)': False, 'has(it)': False, 'has(for)': False, 'has(one)': False, 'has(that)': False, 'has(as)': False, 'has(about)': False, 'has(but)': False, 'has(film)': False, 'has(at)': False, 'has(like)': False, 'has(is)': False, 'bigram(as as)': False, 'bigram(one of)': False, "bigram(it 's)": False, 'bigram(of the)': False, "bigram('s a)": False, 'bigram(the of)': False, 'bigram(to the)': False, 'bigram(the ,)': False}, '2')

**Bigramsets\_with\_preprocessing -**

({'has(the)': False, 'has(,)': False, 'has(a)': True, 'has(of)': False, 'has(and)': False, "has('s)": False, 'has(to)': False, 'has(.)': False, 'has(in)': False, 'has(it)': False, 'has(for)': False, 'has(one)': False, 'has(that)': False, 'has(as)': False, 'has(about)': False, 'has(but)': False, 'has(film)': False, 'has(at)': False, 'has(like)': False, 'has(is)': False, 'bigram(as as)': False, 'bigram(one of)': False, "bigram(it 's)": False, 'bigram(of the)': False, "bigram('s a)": False, 'bigram(the of)': False, 'bigram(to the)': False, 'bigram(the ,)': False}, '2')

**- Negative Features**

**This section includes externally sourced negative word dictionary and the processed version of negative words from prior cleaning. Also included is the processing for whitespaces in some negative words from the initial corpus such as "can't".**

**Negative sets without preprocessing:**

{'has(the)': False,

'has(NOTthe)': False,

'has(,)': True,

'has(NOT,)': False,

'has(a)': False,

'has(NOTa)': False,

'has(of)': False,

'has(NOTof)': False,

'has(and)': False,

'has(NOTand)': False,

"has('s)": False,

"has(NOT's)": False,

'has(to)': False,

'has(NOTto)': False,

'has(.)': False,

'has(NOT.)': False,

'has(in)': False,

'has(NOTin)': False,

'has(it)': False,

'has(NOTit)': False,

'has(for)': False,

'has(NOTfor)': False,

'has(one)': False,

'has(NOTone)': False,

'has(that)': False,

'has(NOTthat)': False,

'has(as)': False,

'has(NOTas)': False,

'has(about)': False,

'has(NOTabout)': False,

'has(but)': False,

'has(NOTbut)': False,

'has(film)': False,

'has(NOTfilm)': False,

'has(at)': False,

'has(NOTat)': False,

'has(like)': False,

'has(NOTlike)': False,

'has(is)': False,

'has(NOTis)': False,

'has(jettisoned)': False,

'has(some)': False,

'has(crucial)': False,

'has(drama)': False}

**STEP 4: EXPERIMENTS**

1. **POS feature**

The default POS tagger (Stanford tagger) was applied on the review to count 4 types of pos tags to use as features, namely: nouns, verbs, adjectives and adverbs.

|  |  |
| --- | --- |
| **POS Sets** | |
| **without preprocessing** | **with preprocessing** |
| {'contains(the)': True,  'contains(,)': True,  'contains(a)': True,  'contains(of)': True,  'contains(and)': True,  "contains('s)": True,  'contains(to)': True,  'contains(.)': True,  'contains(in)': True,  'contains(it)': True,  'contains(for)': True,  'contains(one)': True,  'contains(that)': True,  'contains(as)': True,  'contains(about)': True,  'contains(but)': True,  'contains(film)': True,  'contains(at)': True,  'contains(like)': True,  'contains(is)': True,  'nouns': 186,  'verbs': 94,  'adjectives': 101,  'adverbs': 57} | {'contains(the)': True,  'contains(a)': True,  'contains(of)': True,  'contains(and)': True,  'contains(to)': True,  'contains(in)': True,  'contains(it)': True,  'contains(is)': True,  'contains(s)': True,  'contains(for)': True,  'contains(one)': True,  'contains(that)': True,  'contains(as)': True,  'contains(about)': True,  'contains(film)': True,  'contains(but)': True,  'contains(not)': True,  'contains(at)': True,  'contains(have)': True,  'contains(like)': True,  'nouns': 182,  'verbs': 106,  'adjectives': 100,  'adverbs': 57} |

More past tense verbs mean negative sentiment and more superlative adverb, means positive sentiment so counting POS will also help in sentiment analysis



**Source:** <https://link.springer.com/article/10.1007%2Fs11280-020-00785-z>

1. **Sentiment Lexicon (Subjectivity) feature:**

This is derived from a dictionary that includes subjectivity words from the subjectivity lexicon file (provided) where each word is mapped to a list containing strength and polarity. Negative feature will have number of weakly negative words + 2 \* number of strongly negative words. Same way it will count for positive features. It will not count neutral words

|  |  |
| --- | --- |
| **Subjectivity sets** | |
| **without preprocessing** | **with preprocessing** |
| {'contains(the)': True,  'contains(,)': True,  'contains(a)': True,  'contains(of)': True,  'contains(and)': True,  "contains('s)": True,  'contains(to)': True,  'contains(.)': True,  'contains(in)': True,  'contains(it)': True,  'contains(for)': True,  'contains(one)': True,  'contains(that)': True,  'contains(as)': True,  'contains(about)': True,  'contains(but)': True,  'contains(film)': True,  'contains(at)': True,  'contains(like)': True,  'contains(is)': True,  'positivecount': 336,  'negativecount': 279} | {'contains(the)': True,  'contains(,)': False,  'contains(a)': True,  'contains(of)': True,  'contains(and)': True,  "contains('s)": False,  'contains(to)': True,  'contains(.)': False,  'contains(in)': True,  'contains(it)': True,  'contains(for)': True,  'contains(one)': True,  'contains(that)': True,  'contains(as)': True,  'contains(about)': True,  'contains(but)': True,  'contains(film)': True,  'contains(at)': True,  'contains(like)': True,  'contains(is)': True,  'positivecount': 309,  'negativecount': 273} |

1. **Sentiment Lexicon (LIWC) feature**

I have added pre-processed version of positive words and negative words to their respective dictionary that I got by reading LIWC sentiment lexicon file. For this I have reused function defined for pre-processing of negative words dictionary.

|  |  |
| --- | --- |
| **LIWC sets** | |
| **Without preprocessing** | **With preprocessing** |
| {'contains(the)': True,  'contains(,)': True,  'contains(a)': True,  'contains(of)': True,  'contains(and)': True,  "contains('s)": True,  'contains(to)': True,  'contains(.)': True,  'contains(in)': True,  'contains(it)': True,  'contains(for)': True,  'contains(one)': True,  'contains(that)': True,  'contains(as)': True,  'contains(about)': True,  'contains(but)': True,  'contains(film)': True,  'contains(at)': True,  'contains(like)': True,  'contains(is)': True,  'positivecount': 31,  'negativecount': 26} | {'contains(the)': True,  'contains(,)': False,  'contains(a)': True,  'contains(of)': True,  'contains(and)': True,  "contains('s)": False,  'contains(to)': True,  'contains(.)': False,  'contains(in)': True,  'contains(it)': True,  'contains(for)': True,  'contains(one)': True,  'contains(that)': True,  'contains(as)': True,  'contains(about)': True,  'contains(but)': True,  'contains(film)': True,  'contains(at)': True,  'contains(like)': True,  'contains(is)': True,  'positivecount': 35,  'negativecount': 27} |

**STEP 5: CLASSIFICATION**

**- Using Naïve Bayes classifier**

**Next, the feature extractor is used to process the reviews data and divide the resulting list of feature sets into a training set and a test set.**

**Naïve Bayes classifier is used to train and test data with 80 % of data as training set and 20% as test set initially.**

**The training set is used to train a new "naive Bayes" classifier.**

**Finally, the classifier is examined to determine which features it found most effective for distinguishing the reviews along with a confusion matrix, and summary Accuracy scores (Accuracy, Precision, Recall, F-measure)**

**- Naïve Bayes classification of UNIGRAM sets WITHOUT preprocessing**

Naive Bayes Classifier

Showing most informative features:

Most Informative Features

has(.) = True 0 : 1 = 5.3 : 1.0

has(,) = True 4 : 1 = 5.3 : 1.0

has(.) = False 4 : 0 = 3.1 : 1.0

has(a) = False 3 : 2 = 2.8 : 1.0

has(,) = False 1 : 4 = 1.6 : 1.0

has(a) = True 2 : 3 = 1.3 : 1.0

has(that) = False 4 : 3 = 1.0 : 1.0

has(for) = False 4 : 3 = 1.0 : 1.0

has(the) = False 4 : 3 = 1.0 : 1.0

has(in) = False 4 : 3 = 1.0 : 1.0

None

Confusion matrix:

| 2 3 1 4 0 |

--+------------------------------------+

2 | <56.0%> . . . . |

3 | 20.0% <.> . . . |

1 | 12.0% . <.> . . |

4 | 8.0% . . <.> . |

0 | 4.0% . . . <.>|

--+------------------------------------+

(row = reference; col = test)

Accuracy : 0.56

Precision: 1.0

Recall: 0.2

F-measure: 0.3333333333333333

**- Naïve Bayes classification of UNIGRAM sets WITH preprocessing**

Naive Bayes Classifier

Showing most informative features:

Most Informative Features

has(a) = False 0 : 2 = 2.6 : 1.0

has(a) = True 2 : 0 = 1.3 : 1.0

has(.) = False 4 : 3 = 1.0 : 1.0

has(that) = False 4 : 3 = 1.0 : 1.0

has(for) = False 4 : 3 = 1.0 : 1.0

has(the) = False 4 : 3 = 1.0 : 1.0

has(in) = False 4 : 3 = 1.0 : 1.0

has(but) = False 4 : 3 = 1.0 : 1.0

has(like) = False 4 : 3 = 1.0 : 1.0

has(and) = False 4 : 3 = 1.0 : 1.0

None

Confusion matrix:

| 2 3 1 4 0 |

--+------------------------------------+

2 | <56.0%> . . . . |

3 | 20.0% <.> . . . |

1 | 12.0% . <.> . . |

4 | 8.0% . . <.> . |

0 | 4.0% . . . <.>|

--+------------------------------------+

(row = reference; col = test)

Accuracy : 0.56

Precision: 1.0

Recall: 0.2

F-measure: 0.3333333333333333

From the results, there is no discernible difference in the scores between using Unigram features with or without processing with Naïve Bayes classifier. However, the likelihood of sentiment outcomes vary based on feature token is contained in the parameters.

**- Using Multiple Sci-Kit Learn classifiers to compare with NLTK Naïve Bayes Classifier**

We will also train and test our dataset using 8 algorithms from Sci-kit learner classifiers:

* Random Forest
* MultinomialNB
* BernoulliNB’
* Logistic Regressions
* SGDClassifer
* SVC
* Linear SVC
* NuSVC
* Decision Tree Classifier

|  |  |  |
| --- | --- | --- |
| **CLASSIFIER ACCURACY SCORES** | | |
| **CLASSIFIER** | **Without Preprocessing** | **With Preprocessing** |
| Naïve Bayes | 0.56 | 0.56 |
| MultinomialNB | 0.56 | 0.56 |
| BernoulliNB | 0.56 | 0.56 |
| Decision Tree | 0.6 | 0.56 |
| LogisticRegression | 0.56 | 0.56 |
| SGDCClassifier | 0.4 | 0.12 |
| SVC | 0.56 | 0.56 |
| LinearSVC | 0.56 | 0.56 |
| NuSVC | 0.08 | 0.24 |
| RandomForest | 0.6 | 0.56 |

From the comparison above, Sci-Kit Learn Decisions trees have an accuracy edge over the rest of the classifiers with and without preprocessing of review sentiments. SGDCClassifier has the lowest accuracy scores across the board.

**CROSS VALIDATION**

A 10 fold cross validation using the Naïve Bayes classifier on both processed and unprocessed reviews yielded the following outcomes:

|  |  |
| --- | --- |
| **Naïve Bayes Mean Accuracy with 10 fold cross-validation** | |
| With Processing | 51.4% |
| Without Processing | 51.3% |

**CONCLUSION:**

The Naïve Bayes Model accuracy doesn’t change significantly from the processed features. Preprocessing doesn’t offer considerable leverage over non-processed word features.