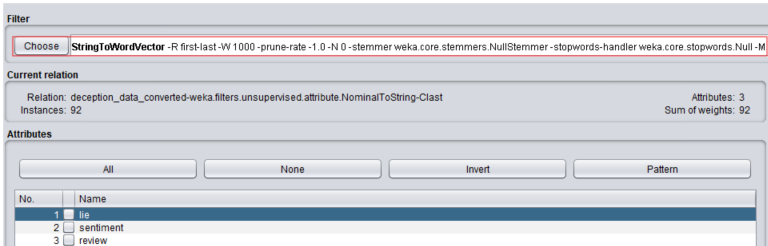
Scenario:   
Some people claimed that machine learning algorithms can figure out whether a person is lying or not. Do you believe that? To test this claim, we have collected a collection of customer reviews, some are true some are fake, and you are going to test how good multinomial NB and SVM scan be for fake review detection.  
This data set also has sentiment label for each review. You will also MNB and SVMs performance in sentiment classification.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Parameter Setting | Overall Accuracy | Precision in Category I | Recall in Category I | Precision in Category II | Recall in Category II |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

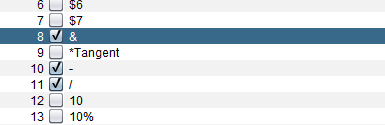
Explain your initial parameter tuning strategy (which parameter to tune, to what option, and theoretical foundation for your choice). Does your strategy help you get better results?  
Compare performance difference in sentiment classification and lie detection, and tell us which task is harder, and try to explain why.   
For each task, use Gain Ratio and Chi2 to rank the features and list top20 features from each method. Based on these top features, can you understand what patterns the classifiers have learned from the data?

Data Pre-Processing:

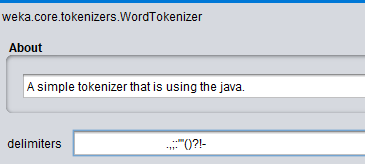
* Import into Weka
* We see three attributes – Lie, Sentiment and review. <review> is our textual data we’ll be focusing on preprocessing and using it for fake review detection and sentiment classification
* Converted the review attribute from string to Word Vector using “StringToWordVector” Filter.

*Preprocessing” panel, click on the Filter “Choose” button, and select “weka->filters->unsupervised->attribute->StringToWordVector”*

* With this filter we get the count of tokens in the review column with additional tuning to perform.
* We noticed meaningless attributes and cleaned (“/” , “&”,”%”)



* Used Tokenizer properties to include “-” to the delimiters

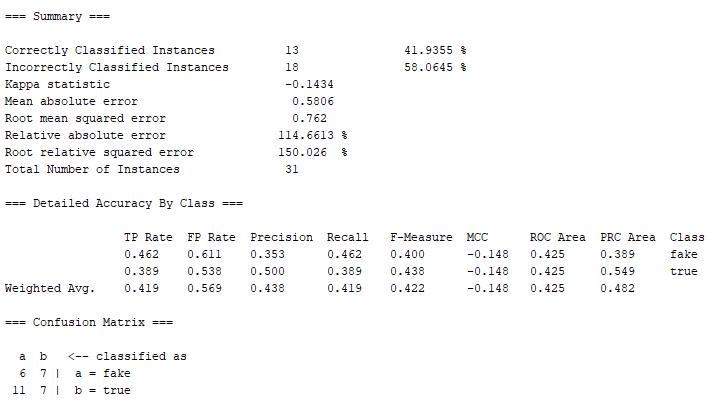


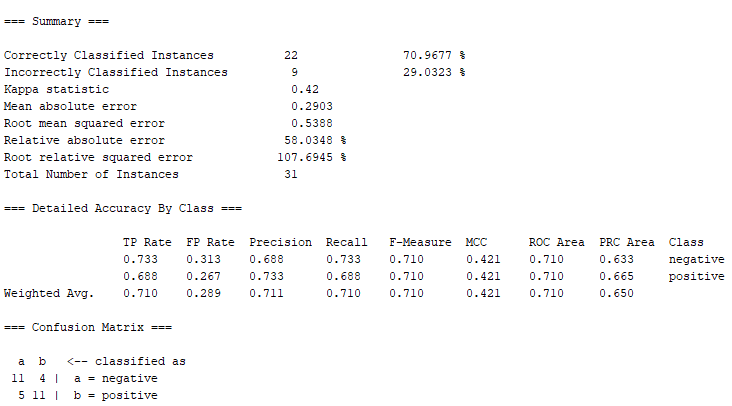
* Reorder attributes for classification.
* Turning on the outputWordCounts to true will provide the raw term frequency (rather than Boolean values).
* Turning on IDFTransform will also weight the IDF.
* Leaving the TFTransform as False, will not turn the term frequency into a log value.
* Defining the attribute indices to just “last”, so that it specifies which attribute we want to apply the vectorization process (the last column, “review”).
* Will also normalize all the data which will normalize the term frequency.
* The lowercase token will be set to True, so that it can merge upper and lower case, by converting all the words to lower case words, and then added to the dictionary.
* Minimum term frequency set to 1, so that it will remove any frequencies that are less than 1. Remove words that only occur once.
* Word to keep parameter is set to 1,000, Weka will sort the words by frequency in each category, and pick the top 1,000 words in each category, and then merge. Instead of this truncation, we can also do this on the entire dataset, we can turn on the “do not operate on per class basis” to True. This will transform the “review” column to many numeric variables, and each variable is a token.

SVM (Support Vector Machine)

* SVM Parameters
  + PolyKernel
  + Calibrator – Linear Regression
  + Split – 66%
    - Lie
    - Sentiment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter Setting** | **Overall Accuracy** | **Precision in Category I** | **Recall in Category I** | **Precision in Category II** | **Recall in Category II** |
| SMO - Lie Detection | 41.93 | 0.353 | 0.462 | 0.500 | 0.389 |
| SMO – Sentiments | 70.96 | 0.688 | 0.733 | 0.733 | 0.688 |





Multi-nominal Naïve Bayes:

Using Naïve Bayes Multi-nominal Text

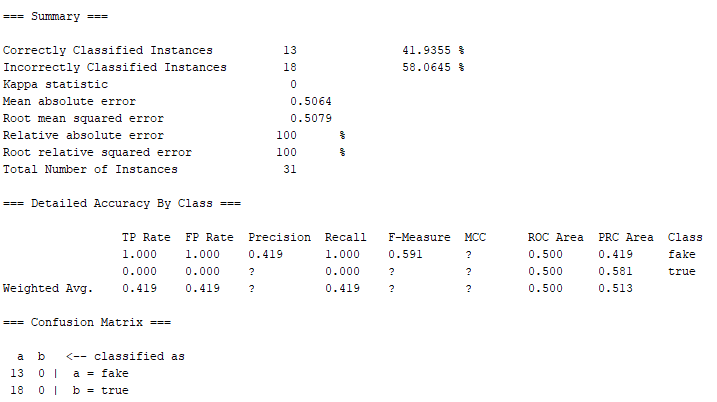
Tuned parameters

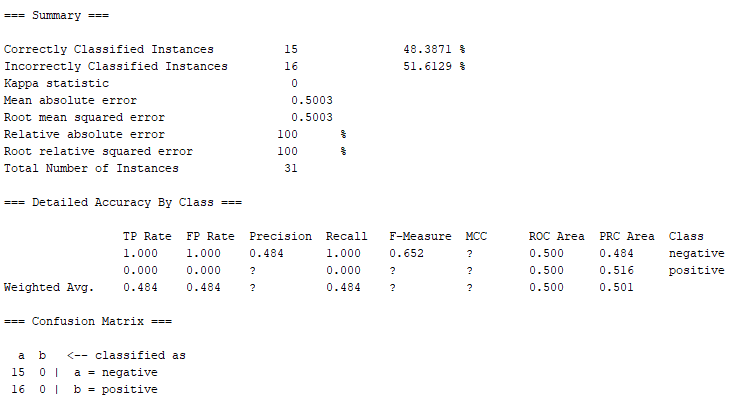
*(Word Tokenizer”-“ to the delimiter. I will keep the stemmer to NullStemmer,*

*Minimum word frequency should be set to 1 and the lower case tokens should be set to True.)*

Output:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Parameter Setting | Overall Accuracy | Precision in Category I | Recall in Category I | Precision in Category II | Recall in Category II |
| MNB / Lie Detection | 41.94 | 0.419 | 1.0 | ? | 0 |
| MNB / Sentiments | 48.39 | 0.484 | 1.0 | ? | 0 |



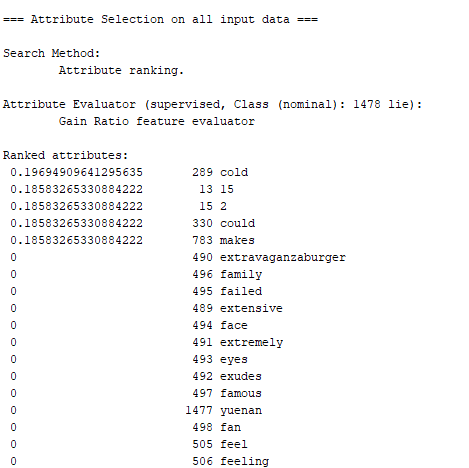


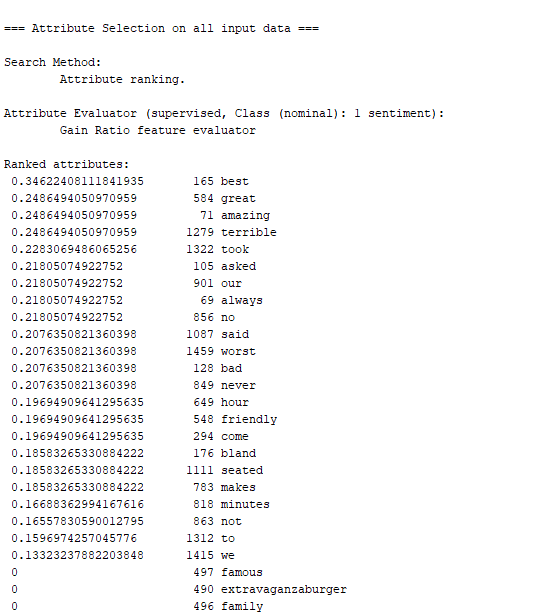
Comparing the above two classifications and lie detection, it seems that for both SVM and MNB, the accuracy rate was higher for sentiment classification. Which also means that sentiment classification is easier than lie detection because sentiment classification looks at the words and possibly weighs them on a scale from negative to positive. Rather for lie detection, just by looking at a word, it is difficult to say if it is true or not, without outside factors like tone of voice.

Gain Ration Attribute evaluation

Want to evaluate the worth of these attributes by measuring the gain ratio. The higher the gain ratio for attributes, the more useful the attribute will be for classification.

Lie Detection





Based on the above, the lie detection not learned as much as the sentiment data. And it obvious that sentiment analysis considered the opposite word like best and terrible, and then has more negative words like bad, worst, and never.