**Data Acquisition**

* Read the data.
* There are 3 different columns. ( Lie, sentiment, review)
* lie and sentiment are nominal values.

**Data Pre processing**

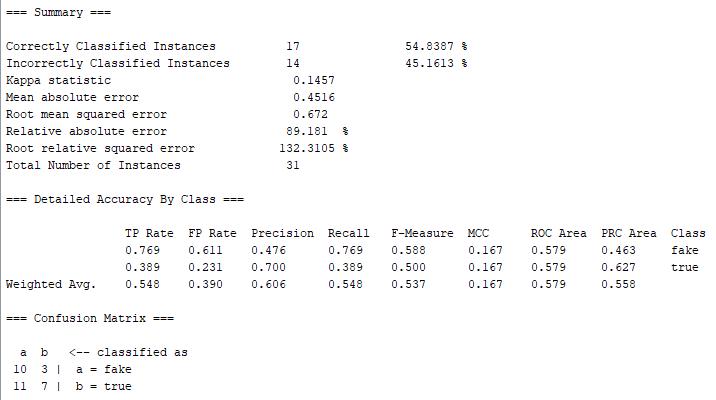
* The review column, however, is not read correctly into Weka
* Hence, use the unsupervised filter in Weka called “StringToWordVector”.
* With this filter, we can count the tokens in the review column with various tuning options. I will use the WordTokenizer which will return individual word tokens.
* Within the word delimiter, I have added the “-“ symbol.
* The stemmer will be NullStemmer which will turn off the stemmer.
* Turning on the outputWordCounts to true will provide the raw term frequency (rather than Boolean values).
* I will also normalize all the data which will normalize the term frequency.
* 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”).
* 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 is a threshold that is set to 1, so that it will remove any frequencies that are less than 1.
* We should remove words that only occur once, because they might be typos.
* Lastly, when the 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.

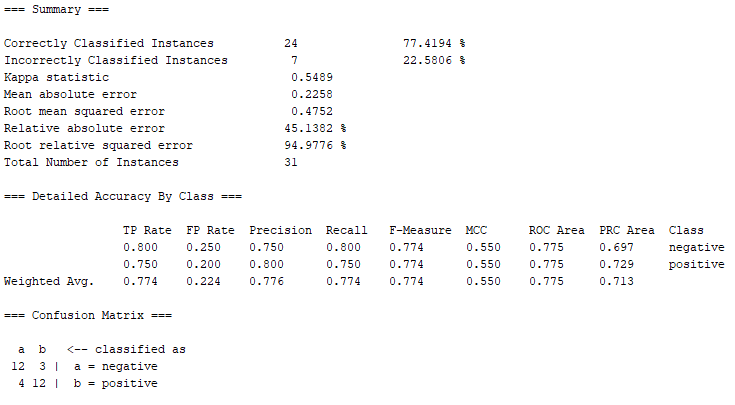
**SVM**

* Now we can tune the SVM parameters.
* Choosing “SMO” under the classifier will open the parameter options.
* Here is where I will use the PolyKernel and choose the Linear Regression calibrator because usually linear kernels work really well with text data.
* We also choose to split the data by 66%, this will help save time. I will first run it on the lie detection variable.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Overall Accuracy** | **Precision in category I** | **Recall in category I** | **Precision in category II** | **Recall in category II** |
| **Lie Detection** | 54.84% | 0.476 | 0.769 | 0.700 | 0.389 |
| **Sentiment** | 77.42% | 0.750 | 0.800 | 0.800 | 0.750 |

Output:



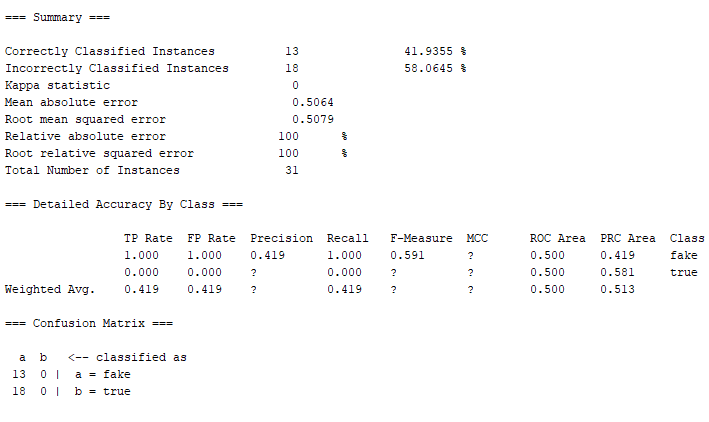


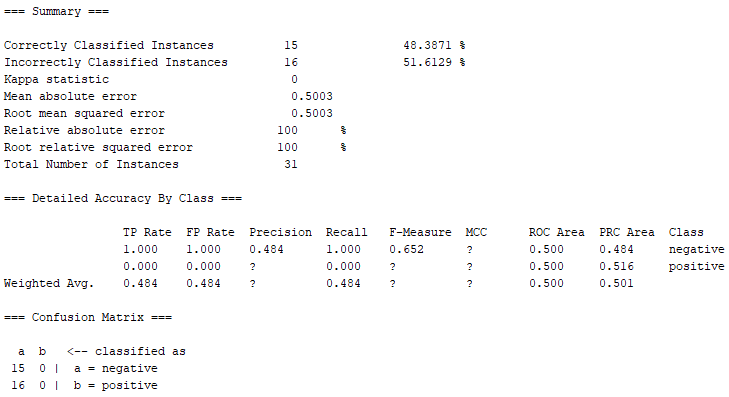
**Multinominal Naïve Bayes:**

* Then I can change the classifier to Naïve Bayes Multinominal Text and tune the parameters.
* I will choose the Word Tokenizer and add ”-“ 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:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Overall Accuracy** | **Precision in category I** | **Recall in category I** | **Precision in category II** | **Recall in category II** |
| **Lie Detection** | 41.94% | 0.419 | 1.000 | ? | 0.000 |
| **Sentiment** | 48.39% | 0.484 | 1.000 | ? | 0.000 |

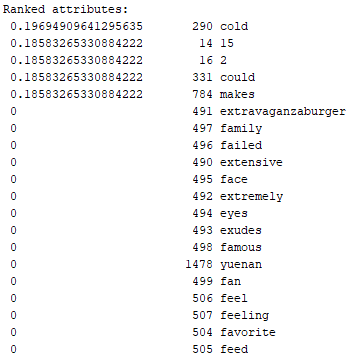


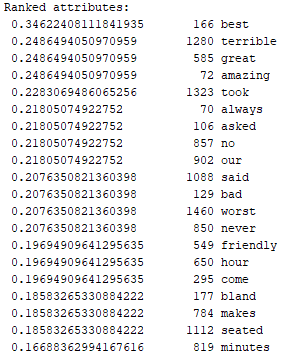


**Comparison:**

Comparing sentiment classification and lie detection, it seems that for both SVM and MNB, the accuracy rate was higher for sentiment classification. I believe 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 Ratio Attribute Evaluate:**

**Lie Detection:**

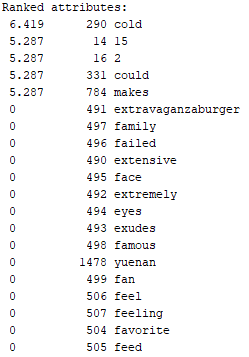
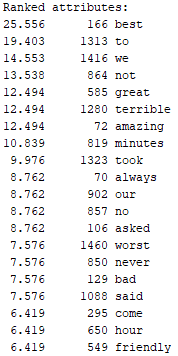
 Sentiment:

The higher the gain ratio for attributes, the more useful the attribute will be for classification.

* The lie detection seems to not have learned as much as the sentiment data.
* It seems that sentiment analysis took into account the opposite word like best and terrible, and then has more negative words like bad, worst, and never.

**ChiSquare Attribute Evaluate:**

Lie detection Sentiment



The Chi squared attribute measures the association between the word feature and its associated class or category. I believe if the attributes have the same ranked number like always, our, no, asked under the sentiment analysis it may mean they are classified together or have some sort of meaning together.