**Capstone Project Report**

**Sentiment Analysis of Movie Reviews**

**Course Number: CKME 136**

**Session: Winter 2016**

**Student Name: Dipak Das**

[**Dipak.das@Ryerson.ca**](mailto:Dipak.das@Ryerson.ca)

**Ryerson University**

TABLE OF CONTENTS

[1.0 Introduction 3](#_Toc447470794)

[2.0 Literature Reviews 3](#_Toc447470795)

[3.0 Dataset 4](#_Toc447470796)

[4.0 Approach 5](#_Toc447470797)

[5.0 Data Collection 6](#_Toc447470798)

[5.1 Training Data 6](#_Toc447470799)

[5.2 Test Data 6](#_Toc447470800)

[6.0 Data Cleansing 6](#_Toc447470801)

[7.0 Processing 7](#_Toc447470802)

[8.0 Implementation of Classifier in Weka 11](#_Toc447470803)

[9.0 Results 11](#_Toc447470804)

[10.0 Conclusion and Discussion 13](#_Toc447470805)

[11.0 Appendix 14](#_Toc447470806)

[12.0 References 22](#_Toc447470807)

# 1.0 Introduction

**Problem**:

Sentiment Analysis is the automated mining of attitudes, opinions, and emotions from text, speech and database sources through Natural Language Processing. The problem here is to analyze the movie review and determine and label the phrases on three categories – positive, neutral and negative.

**Solution:**

The solution is to analyze the sentence and assign the valid label as per each sentiment using three models, namely, Naïve Bayes classification, Support Vector Machine and Random Forest algorithm and compare which method turns out to be best.

# 2.0 Literature Reviews

I reviewed following literatures for the project work:

1. **Mining and Summarizing Customer Reviews**

In the research paper, authors tried to mine and to summarize all the customer reviews of a product. This is different from traditional text summarization because we only mine the features of the product on which the customers have expressed their opinions and whether the opinions are positive or negative. They did not summarize the reviews by selecting a subset or rewrite some of the original sentences from the reviews to capture the main points as in the classic text summarization. The task was performed in three steps: (1) mining product features that have been commented on by customers; (2) identifying opinion sentences in each review and deciding whether each opinion sentence is positive or negative; (3) summarizing the results. Authors proposed a set of techniques for mining and summarizing product reviews based on data mining and natural language processing methods.

Source: <https://www.cs.uic.edu/~liub/publications/kdd04-revSummary.pdf>

1. **Thumbs up? Sentiment Classification using Machine Learning Techniques**

The paper discusses Sentiment Classification of movie review by determining whether a review is positive or negative. Authors describe three machine learning methods (namely, Naïve Bayes, maximum entropy classification and support vector machine), do not perform as well on sentiment classification as on traditional topic-based categorization. In terms of relative performance, Naïve Bayes tends to do the worst and SVMs tend to do the best, although the differences are not very large. On the other hand, they were not able to achieve accuracies on the sentiment classification problem comparable to those reported for standard topic-based categorization.

Source: <http://www.cs.cornell.edu/home/llee/papers/sentiment.pdf>

1. **Text Categorization with Support Vector Machines: Learning with Many Relevant Features**

This paper talks about the benefit of Support Vector Machines (SVM) for text categorization.

It mentions that both theoretical and empirical evidence that SVMs are very well suited for text categorization. The experimental results show that SVMs consistently achieve good performance on text categorization tasks, outperforming existing methods substantially and significantly. SVMs show good performance in all experiments. SVMs do not require any parameter tuning, since they can find good parameter settings automatically. That’s why SVMs a very promising and easy-to-use method for learning text classifiers.

Source: <http://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf>

1. **A Comparative Study of Sentiment Analysis Techniques**

The objective of this paper is to discover the concept of Sentiment Analysis in the field of Natural Language Processing, and the author presents a comparative study of its techniques in this field. Author talks about two main techniques for sentiment analysis: machine learning based and lexicon based. The machine learning approach applicable to sentiment analysis mostly belongs to supervised classification. In unsupervised technique, classification is done by comparing the features of a given text against sentiment lexicons whose sentiment values are determined prior to their use. Author concludes that different types of techniques should be combined in order to overcome their individual drawbacks and benefit from each other’s merits, and enhance the sentiment classification performance.

Source: <https://www.researchgate.net/publication/267764884_A_COMPARATIVE_STUDY_OF_SENTIMENT_ANALYSIS_TECHNIQUES>

# 3.0 Dataset

The source of Dataset of my project is from Kaggle “Sentiment Analysis on Movie Reviews”. The dataset had file for 156,000 records. The data is in CSV format. The details of data is given in Data Collection section

# 4.0 Approach

Input Data from Kaggle

Data Analysis

Use of Dictionary words

Document Term Matrix

Build Random Forest

Build SVM

Build Naïve Bayes Classification

Sentiment Scoring

Compare Models

Results & Conclusion

# 5.0 Data Collection

Data was collected from Kaggle website for Sentiment Analysis on Movie Reviews. It had 156,000 observations of 4 variables, namely, Phrase Id, Sentence Id, Phrase and Sentiment. Each Phrase had a Phrase Id and can have multiple Sentence ID. For example, Sentence Id 1 had 63 phrases, so it had phrase Id 1-63. Each of the phrases has a Sentiment. Each of these phrase is not complete sentences, only one being a complete sentence.

## 5.1 Training Data

In order to train a supervised learning algorithm, a training dataset must be collected. I decided to take 70% of the dataset as training data. Sample of training data (Raw Data) is as follows:

Sample Training Data:

|  |  |  |  |
| --- | --- | --- | --- |
| **PhraseId** | **SentenceId** | **Phrase** | **Sentiment** |
| 64 | 2 | This quiet , introspective and entertaining independent is worth seeking . | 4 |
| 65 | 2 | This quiet , introspective and entertaining independent | 3 |
| 66 | 2 | This | 2 |
| 67 | 2 | quiet , introspective and entertaining independent | 4 |
| 68 | 2 | quiet , introspective and entertaining | 3 |

## 5.2 Test Data

Test data was decided to be 30% of collected data.

# 6.0 Data Cleansing

The aim of the data cleansing process is to remove any unwanted content from Training and Test data. The unwanted content is the information in the phrase that will not be useful for machine learning algorithm. Data cleaning will not only simplify the classification task for the machine learning model but it also will greatly decreasing processing time and cost. I performed the following steps to do the data cleaning:

1. Extracted Phrase and Sentiment from the data (2 variables from the observations)
2. Then I used this output to separate phrases
3. From each of the phrases, I filtered the phrases that are complete sentence. This was done by selecting only the phrases that started with capital letter and ended with period.
4. These set of data (which are complete sentences) will be used to determine the sentiments.
5. For these sentences, I stripped white spaces, punctuations and English stop words.
6. In the end, I created Term Document Matrix for training and test data which are used subsequently to find the sentiment scores and build the models.

**Positive and Negative dictionary**

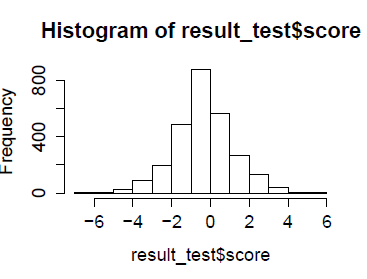
I used Hu and Liu’s “opinion lexicon” categorizes nearly 6,800 words as positive or negative and can be downloaded from Bing Liu’s web site: <http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar>

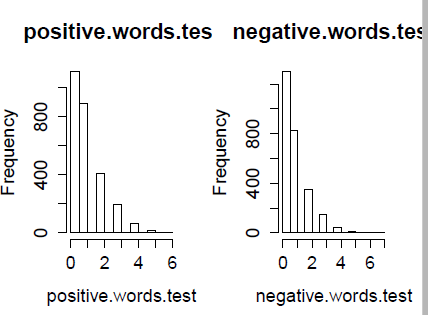
The lexicon consists of two text files, one containing a list of positive words and the other containing negative words. Each file begins with some documentation, which we need to skip and is denoted by initial semi-colon (“;”) characters.

# 7.0 Processing

* After doing data cleansing, used the sentences to find the total words, positive words, negative words and other words from each sentence with the help of dictionary words as mentioned above
* Then, for each sentence found out % of Positive and & of Negative words
* In the next step, I used a scoring algorithm to find the scores of each sentences, namely, 0 for Neutral, <0 for Negative and >0 for Positive words
* I did some histogram based on this scores for Training and Test data as follows:

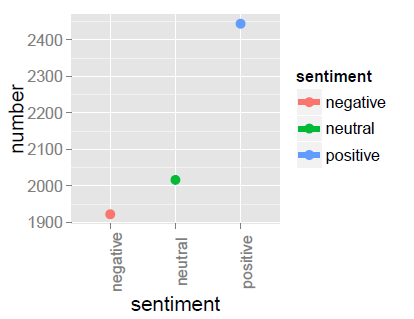




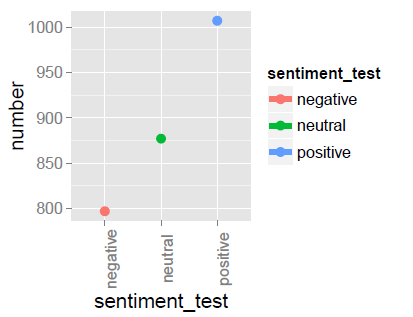


I did the visualization of Positive, Negative and Neutral sentiments for Training and Test data. Here are the graphs:

**Training Data**

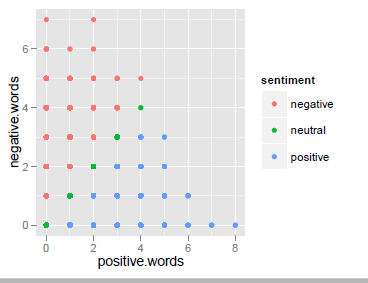


**Test Data**

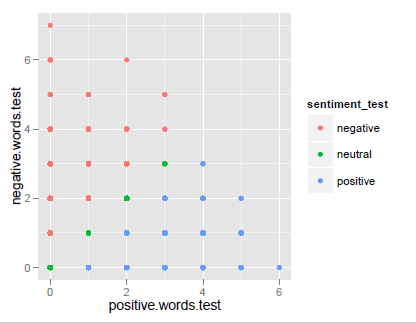


Here are the qplot for Training and Test data:

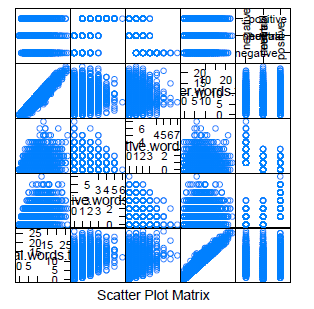
**Training Data**



**Test Data**



Here is the Scatter Plot Matrix:



# 8.0 Implementation of Classifier in Weka

I decided to use Weka as it offered several benefits e.g., ease of use, large library of classification algorithms and good visualization. I tried three models – Naïve Bayes, Support Vector Machine and Random forest.

All models were evaluated using 10-fold cross validation. Each model was evaluated using several criteria: accuracy, precision, recall, mean absolute error, RMSE, correctly classified percentage and system time.

Confusion Matrix was created for each model. Precision and Recall from the results of Confusion Matrix.

For all 3 models, training data was used to train the model. Once the trained model was successful, test data was used to fit the model. Evaluate Weka Classifier was done for each of the models.

# 9.0 Results

Accuracy

The Accuracy is the percentage of correctly classified instances.

Precision

Precision measures the exactness of a classifier. A higher precision means less false positives, while a lower precision means more false positives. An easy way to improve precision is to decrease recall.

Recall

Recall measures the completeness, or sensitivity, of a classifier. Higher recall means less false negatives, while recall means more false negatives. Improving recall can often decrease precision because it gets increasingly harder to be precise as the sample increases.

F-measure

The F-score measure gives a good indication of the overall performance of a classifier and is calculated using the formula:

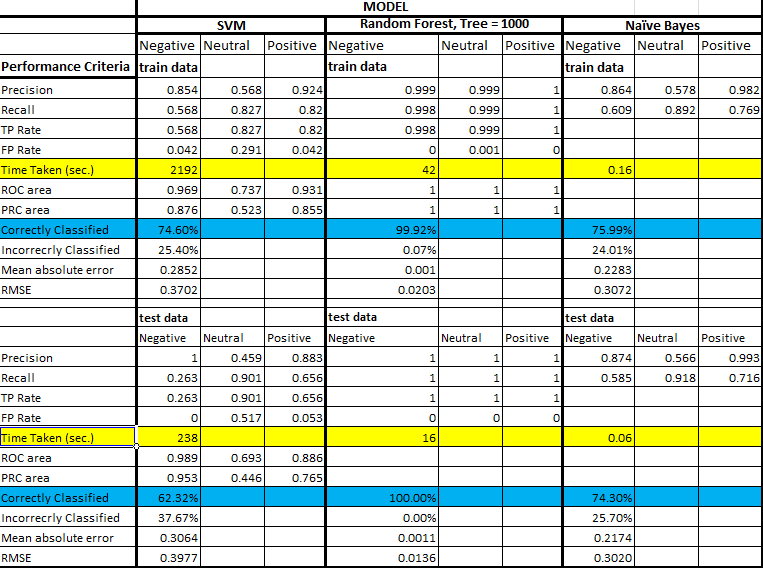
F-measure = 2(Precision)(Recall) / Precision + Recall

System Time

For each of the models, system time is noted for performing the calculation and this was compared to come to decision for the model to be selected.

Here is the performance matrix for all 3 models as applied to the Training and Test data:

**Performance Matrix**



# 10.0 Conclusion and Discussion

SVM Performance:

For Training data, SVM correctly classified 75% of the observation and rest 25% were classified incorrectly.

For Test data, SVM correctly classified only 62% of the observations

SVM took maximum time to process the data among all 3 models. It took 200 seconds to process test data. One possible reason for SVM to take much longer time is probably adding extra dimensions to the size feature space in an attempt to separate the classes.

Naïve Bayes Performance:

For Training data, NB classified slightly better than SVM. It correctly classified 76% of the observations.

For Test data, NB classified 74% observations correctly. Naïve Bayes took lot less time to process, around 0.06 seconds for test data

Random Forest Performance:

I did two iterations of Random Forest model. First iterations was with Tree = 1000 and the second iteration was with Tree = 500.

For Tree = 1000 & Tree = 500, with **Training data** there were 0.0783% incorrect classifications. These were same during cross validation

For Tree = 1000, with **Test data** there were 0.0313% incorrect classifications.

However, for **Tree = 500**, with Test data **100% of observations** were classified correctly.

Random Forest model took about 42 seconds to run for Tree = 1000 and about 22 seconds to run for Tree = 500 both for Train data. This was much better than SVM which took about 2200 seconds to run for Train data.

As random forest used ensemble model, it gave much improved performance than that of a singular model. It reduces the variance which probably increased the overall performance as compared to the other two models.

On the basis of all the performance criteria, **Random Forest model is the best model** that is applied to this data and this model is selected for the project

# 11.0 Appendix

**Support Vector Machine Results:**

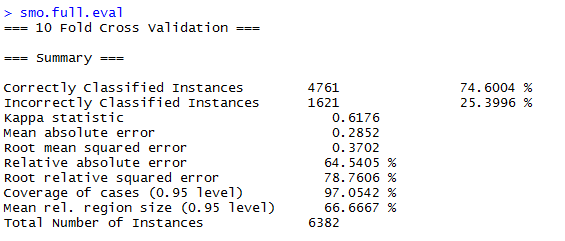
**Train Data:**

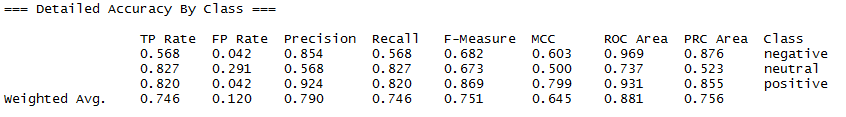
smo.full.eval <- evaluate\_Weka\_classifier(smo.full, numFolds=10,train.final, class=T)

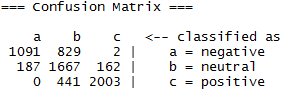
> system.time(smo.full.eval <- evaluate\_Weka\_classifier(smo.full, numFolds=10,train.final\_1, class=T))

user system elapsed

2181.76 0.55 2192.14







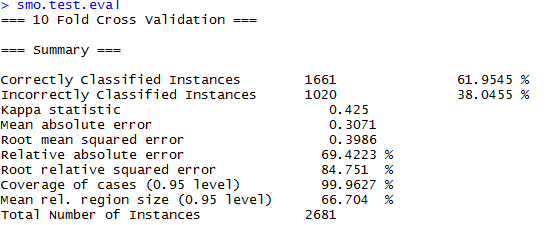
**Test Data:**

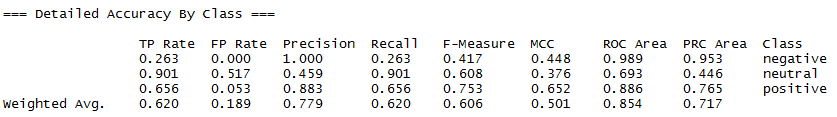
> system.time(smo.test.eval <- evaluate\_Weka\_classifier(smo.full, numFolds=10,test.final\_svm, class=T))

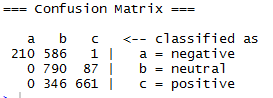
user system elapsed

236.74 0.04 237.70

>







**Random Forest Results:**

Random Forest Train data

**Tree size = 1000**

|  |
| --- |
| > rf <- make\_Weka\_classifier("weka/classifiers/trees/RandomForest")  > system.time(randf.full <- rf(sentiment\_factor ~ positive.words + negative.words + other.words, data=train.final\_1, control = Weka\_control(I = 1000)))  user system elapsed  7.33 0.16 6.94  > summary(randf.full)  === Summary ===  Correctly Classified Instances 6380 99.9687 %  Incorrectly Classified Instances 2 0.0313 %  Kappa statistic 0.9995  Mean absolute error 0.0006  Root mean squared error 0.0142  Relative absolute error 0.1255 %  Root relative squared error 3.0141 %  Coverage of cases (0.95 level) 100 %  Mean rel. region size (0.95 level) 33.4482 %  Total Number of Instances 6382  === Confusion Matrix ===  a b c <-- classified as  1920 2 0 | a = negative  0 2016 0 | b = neutral  0 0 2444 | c = positive |
|  |
| |  | | --- | | > | |

> print(randf.full)

Random forest of 1000 trees, each constructed while considering 2 random features.

Out of bag error: 0.0008

> system.time(randf.full.eval <- evaluate\_Weka\_classifier(randf.full, numFolds=10,train.final\_1, class=T) )

user system elapsed

43.44 0.08 42.29

|  |
| --- |
| > randf.full.eval  === 10 Fold Cross Validation ===  === Summary ===  Correctly Classified Instances 6377 99.9217 %  Incorrectly Classified Instances 5 0.0783 %  Kappa statistic 0.9988  Mean absolute error 0.0011  Root mean squared error 0.0207  Relative absolute error 0.2523 %  Root relative squared error 4.4042 %  Coverage of cases (0.95 level) 100 %  Mean rel. region size (0.95 level) 33.5527 %  Total Number of Instances 6382    === Confusion Matrix ===  a b c <-- classified as  1919 3 0 | a = negative  2 2014 0 | b = neutral  0 0 2444 | c = positive |
|  |
| |  | | --- | | > | |

Random Forest Test data

**Tree size = 1000**

> system.time(randf.test.eval <- evaluate\_Weka\_classifier(randf.full, numFolds=10,test.final.rf, class=T))

user system elapsed

16.92 0.00 16.40

|  |
| --- |
| > randf.test.eval  === 10 Fold Cross Validation ===  === Summary ===  Correctly Classified Instances 2679 99.9254 %  Incorrectly Classified Instances 2 0.0746 %  Kappa statistic 0.9989  Mean absolute error 0.0013  Root mean squared error 0.0176  Relative absolute error 0.2922 %  Root relative squared error 3.7319 %  Coverage of cases (0.95 level) 100 %  Mean rel. region size (0.95 level) 33.6193 %  Total Number of Instances 2681    === Confusion Matrix ===  a b c <-- classified as  797 0 0 | a = negative  0 877 0 | b = neutral  0 2 1005 | c = positive |
|  |
| |  | | --- | | > | |

Random Forest Train data

**Tree size = 500**

> system.time(randf.full2 <- rf(sentiment\_factor ~ positive.words + negative.words + other.words, data=train.final\_1, control = Weka\_control(I = 500)))

user system elapsed

4.82 0.24 4.79

> summary(randf.full2)

=== Summary ===

Correctly Classified Instances 6380 99.9687 %

Incorrectly Classified Instances 2 0.0313 %

Kappa statistic 0.9995

Mean absolute error 0.0006

Root mean squared error 0.0143

Relative absolute error 0.1258 %

Root relative squared error 3.0382 %

Coverage of cases (0.95 level) 100 %

Mean rel. region size (0.95 level) 33.4482 %

Total Number of Instances 6382

=== Confusion Matrix ===

a b c <-- classified as

1920 2 0 | a = negative

0 2016 0 | b = neutral

0 0 2444 | c = positive

> system.time(randf.full.eval2 <- evaluate\_Weka\_classifier(randf.full2, numFolds=10,train.final\_1, class=T) )

user system elapsed

23.50 0.15 22.45

>

> randf.full.eval2

=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances 6377 99.9217 %

Incorrectly Classified Instances 5 0.0783 %

Kappa statistic 0.9988

Mean absolute error 0.001

Root mean squared error 0.0186

Relative absolute error 0.2215 %

Root relative squared error 3.9475 %

Coverage of cases (0.95 level) 100 %

Mean rel. region size (0.95 level) 33.5475 %

Total Number of Instances 6382

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

0.998 0.000 0.999 0.998 0.999 0.998 1.000 1.000 negative

0.999 0.001 0.999 0.999 0.999 0.998 1.000 1.000 neutral

1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 positive

Weighted Avg. 0.999 0.000 0.999 0.999 0.999 0.999 1.000 1.000

=== Confusion Matrix ===

a b c <-- classified as

1919 3 0 | a = negative

2 2014 0 | b = neutral

0 0 2444 | c = positive

Random Forest Test data

**Tree size = 500**

> system.time(randf.test.eval2 <- evaluate\_Weka\_classifier(randf.full2, numFolds=10,test.final.rf, class=T))

user system elapsed

8.76 0.02 8.33

> randf.test.eval2

=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances 2681 100 %

Incorrectly Classified Instances 0 0 %

Kappa statistic 1

Mean absolute error 0.0012

Root mean squared error 0.0138

Relative absolute error 0.264 %

Root relative squared error 2.9293 %

Coverage of cases (0.95 level) 100 %

Mean rel. region size (0.95 level) 33.6069 %

Total Number of Instances 2681

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 negative

1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 neutral

1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 positive

Weighted Avg. 1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000

=== Confusion Matrix ===

a b c <-- classified as

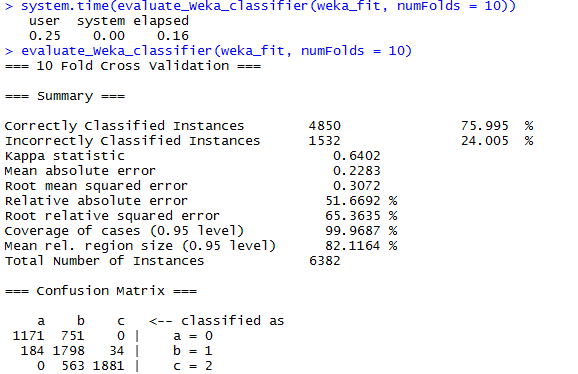
797 0 0 | a = negative

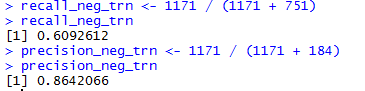
0 877 0 | b = neutral

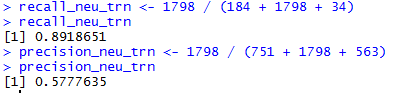
0 0 1007 | c = positive

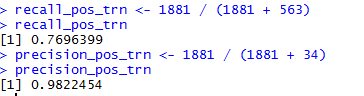
**Naïve Bayes Results:**

**Training Result:**



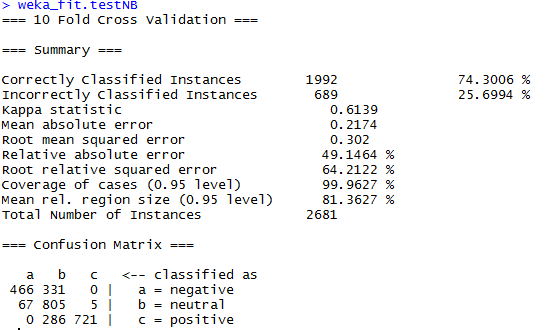


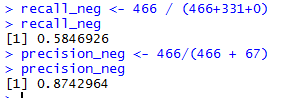


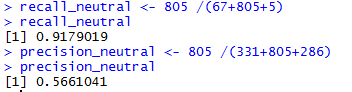


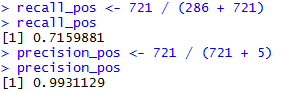
**Test Result:**











# 12.0 References

1. R for Medicine and Biology – by Paul D Lewis
2. Modeling Techniques in Predictive Analytics with Python and R: A Guide to Data Science – by Thomas W Miller
3. Opinion Lexicon – Hu and Liu <https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html>
4. Predictive Analytics for Dummies
5. Sentiment analysis algorithms and applications: A Survey – Walaa Medhat, Ahmed Hassan, Hoda Korashy
6. Classification and Regression by Random Forest - Andy Liaw and Matthew Wiener
7. Mining Twitter for Airline Consumer Sentiment by Jeffrey Bean

<http://www.inside-r.org/howto/mining-twitter-airline-consumer-sentiment>

1. First shot: Sentiment Analysis in R by Andy Bromberg

<http://andybromberg.com/sentiment-analysis/>

1. Sentiment analysis with Machine Learning in R – Timothy P. Jurka