IDS 572 Assignment 5

Aditi Vishwasrao – 674849041

Mounica Sirineni - 652518967

Nishanth Reddy Konkala - 658824939

**1. Read a few of the movie-reviews. If you were to classify manually, are there certain words/terms indicative of ‘positive’ and ‘negative’ that you would use? Provide lists (‘dictionaries’) of ‘positive’ and ‘negative’ terms. Now consider a manual classification process – you may count the number of ‘positive’ and ‘negative’ terms (matching your dictionaries) and classify based on majority count. How effective is this (accuracy?)?**

**Answer:** We went through a few movie reviews and prepared a list of terms that indicate a positive or negative review. The complete list of terms are as follows:





A few of the positive reviews are: strong, extraordinary, interesting, good, special, strong, likeable,etc.

A few of the negative reviews are: aweful, horrific, unfunny, worst, mess, boring, slow, lame, etc.

The list of files that were considered for manual prediction is as follows:



We get the following confusion matrix from the results:

Accuracy: 55%

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted Positive | Predicted Negative | Precision |
| Actual Positive | 15 | 5 | 75% |
| Actual Negative | 13 | 7 | 35% |
| Recall | 70% | 30% |  |

**Conclusion:**In this method of manual prediction, the accuracy is 55%. This is not very good accuracy for a model. Also, the number of negative cases that have been detected accurately are very less. We also need to take into consideration that we have taken into account a very few (20 each) positive and negative reviews for classification. We can conclude that the manual process is not very effective.

**2. Using automated text-mining: perform the steps noted above to get a document-term matrix. How many terms do you have? With and without stemming?**

**Answer:** We performed automated text mining using tokenization, filtering stop words, transforming cases and filtering tokens by length. We evaluated for stemming. The table provides a summary. [Click here](#auto) to view the screenshots.

**Table 2.1 Automated text mining**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Number of Examples** | **Number of Special Attributes** | **Number of Regular Attributes** |
| Without Stemming | 2000 | 6 | 2425 |
| With Stemming | 2000 | 6 | 1040 |

**Conclusion:** From the table, we observe that stemming decreases the number of regular attributes. From the definition of stemming, we say that stemming reduces the words to its root words.

**3. The sentiment polarity of a document provides the ‘label’/target variables. Learn knn and naïve- Bayes models to classify for sentiment polarity. To evaluate effectiveness in prediction, we will consider Split-Validation, with 50% of the documents used for training and 50% for test (you can set a local random number seed for the Split-Validation operator to get the same split on the data across experiments).**

**(a) Develop and evaluate knn and naïve-Bayes models. What values of k work best? Evaluation is based on the confusion matrix – overall accuracy, accuracies on positive and negative, precision, specificity, sensitivity, etc.**

**Answer:** We have developed KNN and Naïve Bayes models. Below table provides a summary of performance. [Click here](#knnnb) to view the screenshots of performance vector.

**Table 3.1 KNN and Naïve Bayes models**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Dataset** | **Overall accuracy** | **Class recall (true negative)** | **Class recall (true positive)** |
| KNN with K=3 | Training data | 80.80% | 79.43% | 82.12% |
| Testing data | 67% | 61.10% | 73.12% |
| KNN with K=4 | Training data | 77.40% | 88.80% | 66.40% |
| Testing data | 65.80% | 73.48% | 57.84% |
| KNN with K=5 | Training data | 78.10% | 75.15% | 80.94% |
| Testing data | 64.7% | 57.56% | 72.1% |
| KNN with K=10 | Training data | 72.30% | 78% | 66.80% |
| Testing data | 66.40% | 70.92% | 61.71% |
| KNN with K=30 | Training data | 73.30% | 79.63% | 67.19% |
| Testing data | 67.40% | 72.30% | 62.32% |
| Naïve Bayes | Training data | 100% | 100% | 100% |
| Testing data | 64.90% | 59.53% | 70.47% |

**Conclusion:** From the above table, we observe that **KNN with K = 4** is the best model. Further analysis is done using this model.

**You should experiment with the following to determine how they affects performance:**

**(i) measures defining the document term matrix – binary term occurrence, term occurrence, term frequency, tf-idf (please define these four term as part of the assignment)**

**Answer:**KNN with K = 4 model is evaluated using different measures. The table below provides a summary of performances. [Click here](#measures) to view the screenshots of performance vector.

Binary term occurrence: If a word occurs in the document, then the vector is “1” otherwise “0”.

Term occurrence: It gives the number of times the term occurs in the document.

Term frequency: It gives the frequency of the term in the document.

TF-IDF: (Term Frequency – Inverse Document Frequency) It indicates the frequency of a term in the document and in the corpus of documents.

**Table 3.2 KNN with K = 4 and different measures**

|  |  |  |
| --- | --- | --- |
| **Document-term matrix** | **Training accuracy** | **Testing accuracy** |
| Binary term occurrence | 95.10% | 51.80% |
| Term occurrence | 71.3% | 52.6% |
| Term frequency | 74.20% | 63.80% |
| TF-IDF | 77.40% | 65.80% |

**Conclusion:** From the above table, we observe that KNN with K=4 and TF-IDF measure is the best model.

**(ii) which words to include**

**- without and with stemming**

**Answer:** KNN model with K = 4 and TF-IDf measure is evaluated on stemming. [Click here](#stemming) to view the screenshots of performance vector.

**Table 3.3 KNN with K = 4, TF-IDf, with and without stemming**

|  |  |  |
| --- | --- | --- |
| **Document-term matrix** | **Training accuracy** | **Testing accuracy** |
| With stemming | 76.80% | 67% |
| Without stemming | 77.40% | 65.80% |

**Conclusion:**From the above table, we observe that KNN model with K=4, TF-IDF measure and with stemming is the best model.

**- using words that match specific POS tags – you need to determine which POS tags you want to include (there are multiple websites that give the list of POS tags)**

**Answer:** KNN model with K = 4 , TF-IDF measure and stemming is evaluated for POS tags. We have used two different combinations of POS tags. [Click here](#pos) to view the screenshots of performance vector.

**Table 3.4 KNN with K = 4, TF-IDf, with stemming and POS tags**

|  |  |  |
| --- | --- | --- |
| **POS tags** | **Training accuracy** | **Testing accuracy** |
| JJ.\*|NN.\*|RB.\*|VB.\* | 76.40% | 66.60% |
| VB.\*|NN.\*|RB.\* | 76.90% | 63.70% |

From the above table, we observe that the best POS tags are combination of adjective, noun, adverb and verb. Now we will evaluate whether this best POS tags are helping the model or not in terms of accuracy.

**Table 3.5 KNN with K = 4, TF-IDf, with stemming, with and without POS tags**

|  |  |  |
| --- | --- | --- |
| **POS tags** | **Training accuracy** | **Testing accuracy** |
| With POS tags | 76.40% | 66.60% |
| Without POS tags | 76.80% | 67% |

From the above table, we observe that model has better accuracy without using any POS tags.

**- pruning words based on min and max of the number of documents they appear in**

**Answer:** KNN model with K = 4, TF-IDF measure, stemming and without POS tags is evaluated for pruning. We have used min of 10 and max of 100 documents for pruning. [Click here](#pruning) to view the screenshots of performance vector.

**Table 3.6 KNN with K = 4, TF-IDf, with stemming, without POS tags, with and without pruning**

|  |  |  |
| --- | --- | --- |
| **Pruning** | **Training accuracy** | **Testing accuracy** |
| With pruning | 75.90% | 64.90% |
| Without pruning | 76.80% | 67% |

From the above table, we observe that **KNN withK = 4, TF-IDf, with stemming, without POS tags and without pruning** is the best model.

**(b) For any one setting from above (for example, using tf-idf, stemmed terms, and using some reasonable pruning), build a support vector machine model. Try SVM with dot-kernel. How does the performance of the SVM model compare with k-nn and naïve-Bayes models? Does using a radial basis function kernel (nonlinear model) perform better? Also try a random forest model and compare performance. What do you conclude overall?**

**Answer:** We have used our best model setting from previous question. TF-IDF measure with stemming, without POS tags and without pruning. We developed SVM. Random forest, KNN with K = 4 and Naïve Bayes for this model. [Click here](#models) to view the screenshots of performance vectors.

**Table 3.7 Different models**

|  |  |  |
| --- | --- | --- |
| **Model** | **Training accuracy** | **Testing accuracy** |
| SVM with dot-kernel | 100% | 77.50% |
| SVM with radial-kernel | 50.90% | 49.10% |
| Random Forest | 52.30% | 49.60% |
| KNN with K = 4 | 76.80% | 67% |
| Naïve Bayes | 99.90% | 64.90% |

**Conclusion:** From the above table, we observe that SVM with dot-kernel performs better than radial kernel. When we compare all models, KNN with K = 4 is the best model.

**(c) Use the Harvard-IV dictionary of positive and negative terms - Filter Tokens by Content to keep only these terms. How many terms do you get? (Note: for this, the document terms should not be stemmed, since they would otherwise not match the unstemmed dictionary terms. Does it make sense to use pruning of terms or POS tags here?). Use just the count of positive and negative terms to classify sentiment of reviews. How do you do this, and what performance do you find? Develop and evaluate a naïve-Bayes, a SVM (either with dot kernel or other kernel based on what you found to perform better in part (b) above), and a random forest model, using only the positive and negative words.**

**Answer:**

By filtering tokens by content of Harvard-IV dictionary of positive and negative terms, we get 2885 terms. Pruning or using POS tags may result in different token from the review documents which will not match the Harvard IV dictionary terms. Thus, pruning or using POS tags does not make sense.

[Click here](#harvard) to view the screenshots of performance vector.

**Table 3.8 Harvard dictionary with different models**

|  |  |  |
| --- | --- | --- |
| **Model** | **Training accuracy** | **Testing accuracy** |
| Naïve Bayes | 93.20% | 57.60% |
| SVM with dot-kernel | 97.20% | 67.10% |
| Random Forest | 55.90% | 53.40% |
| KNN with K = 4 | 75.90% | 61.00% |

We used only the SVM with dot kernel model here as this was the better of the two models used for part (b). From the above table, we can see that KNN with K=4 is clearly the best model.

**(d) Next, use Wordnet and Senti-Wordnet to determine the sentiment polarity of the reviews. Provide a description of how this process works – i.e. how WordNet and Senti-Wordnet operates. Use the Senti-WordNet sentiment scores to classify the reviews. What performance do you find?**

**Answer:**

*“Word net is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Sysnets are interlinked by means of conceptual semantic and lexical relations” ~* **Princeton University WordNet***.*

Whereas Senti-WordNet is a lexical resource for opinion mining. It assigns each synset of Word Net three sentiment scores:Positivity, negativity, and objectivity.

[Click here](#wordnet) to view the screenshots of performance vector.

**Table 3.9 Wordnet dictionary with different models**

|  |  |  |
| --- | --- | --- |
| **Model** | **Training accuracy** | **Testing accuracy** |
| Naïve Bayes | 100% | 63% |
| SVM with dot-kernel | 100% | 75.60% |
| Random Forest | 50.90% | 49.10% |
| KNN with K = 4 | 78% | 64.90% |

**Conclusion:** KNN with K = 4 is the best model.

**(e) How does performance in (a) - (d) above compare? What can you conclude? Does the smaller list of ‘sentiment’ words provide similar performance levels as models built on a broader set of terms? How does the performance of Senti-WordNet compare? Does it help to include Senti-WordNet scores in a model with sentiment-words or with the broader set of terms?Which is your ‘best’ model?**

KNN with k=4 yields the best model in all cases from (a) to (d). Comparing the training and validation accuracies in all four cases, we do not find any significant differences in the performance of the KNN model in each of the four cases.

However, considering only the validation accuracy, the smaller list of sentiment words looks to be giving a slightly better performance.

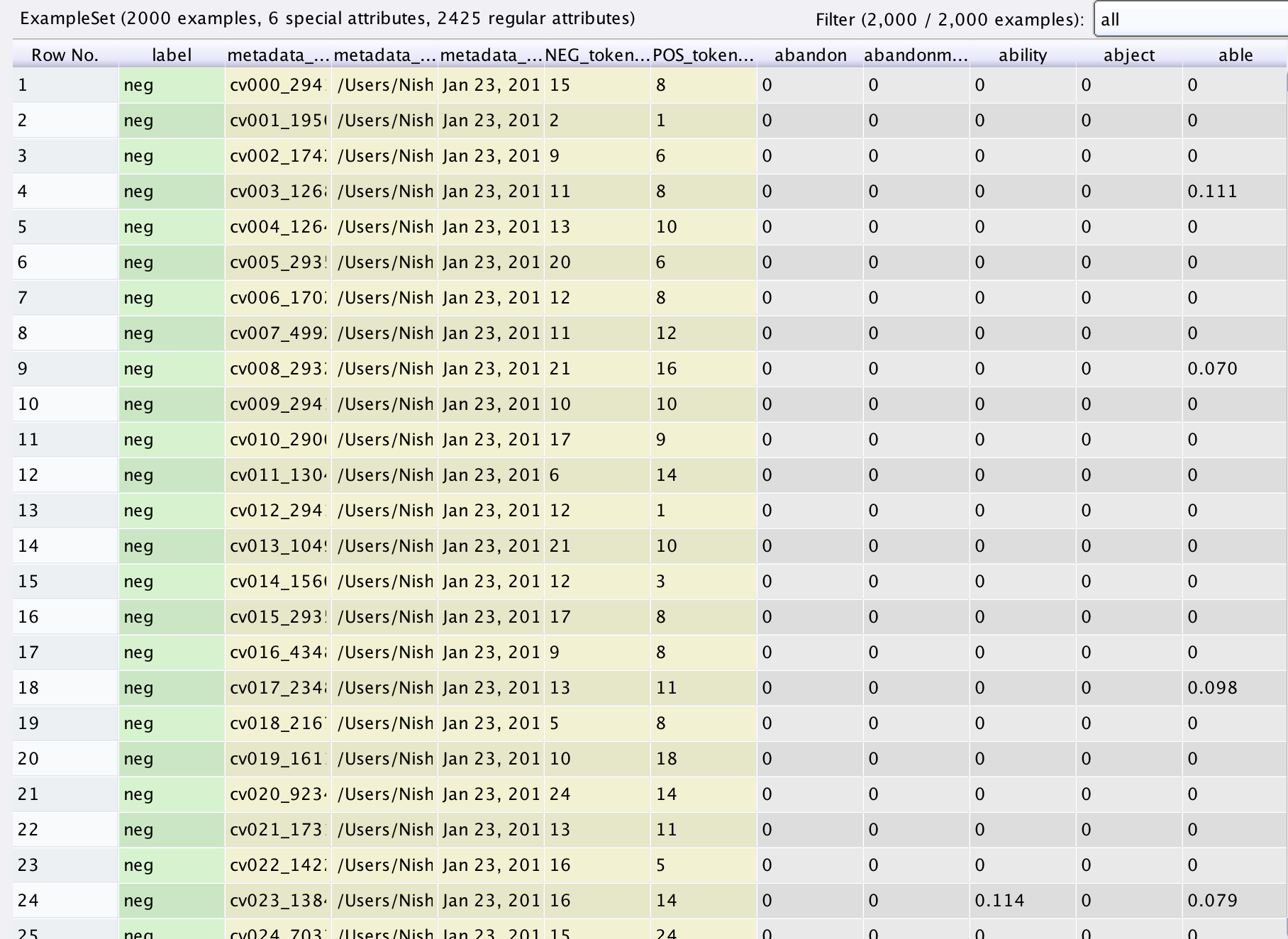
The inclusion of Senti-Wordnet scores does not seem to help to a great extent.

Our best model is KNN with k=4 with stemming, but without using POS tags or pruning.

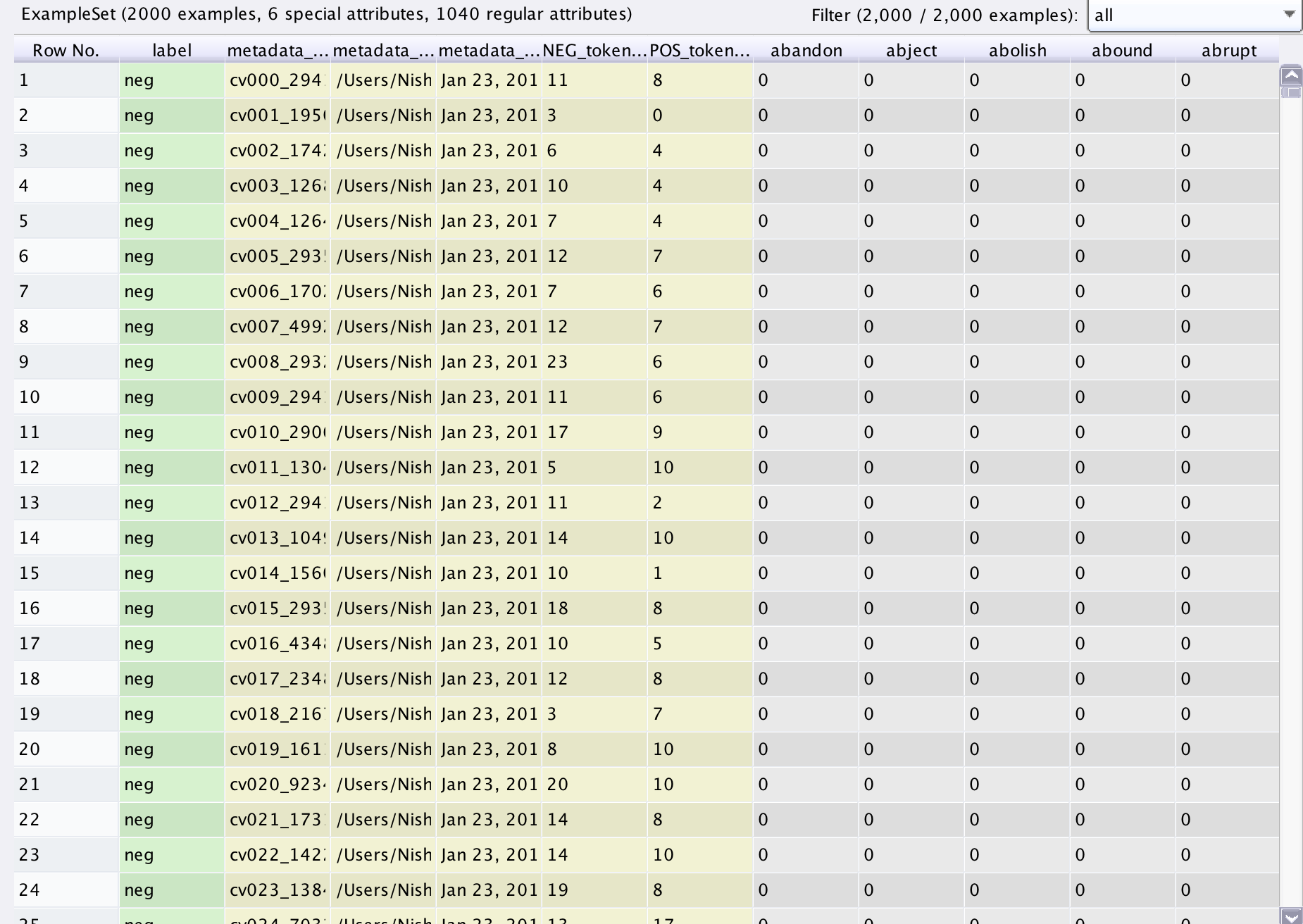
**4. Appendix:**

**4.1 Automated text mining** [**[Back to the top]**](#t21)

**4.1.1 Without stemming**

****

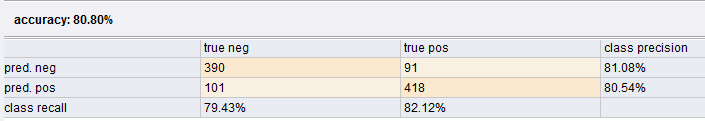
**4.1.2 With stemming**

****

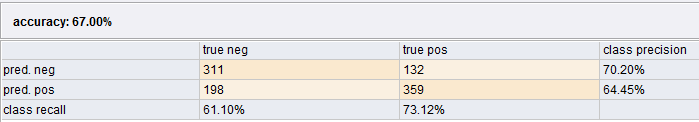
**4.2 KNN and Naïve Bayes models** [**[Back to the top]**](#t31)

**4.2.1 K = 3**

**Training data**

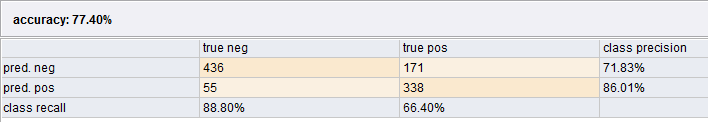
****

**Testing data**

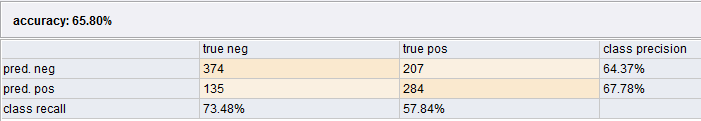
****

**4.2.2 K = 4**

**Training data**

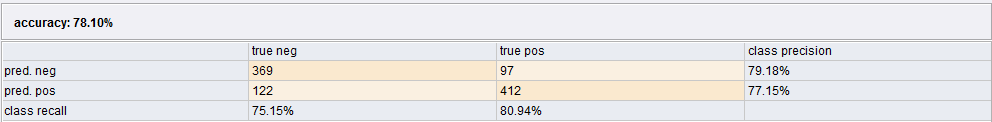
****

**Testing data**

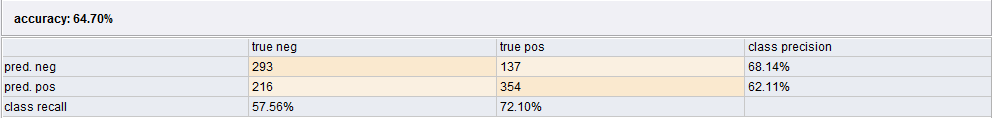
****

**4.2.3 K = 5**

**Training data:**

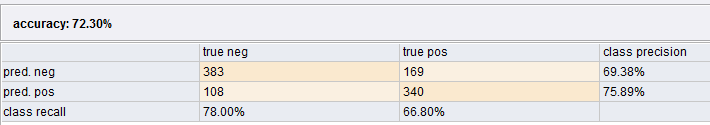
****

**Testing data:**

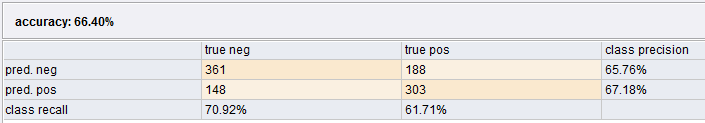
****

**4.2.4 K = 10**

**Training data**

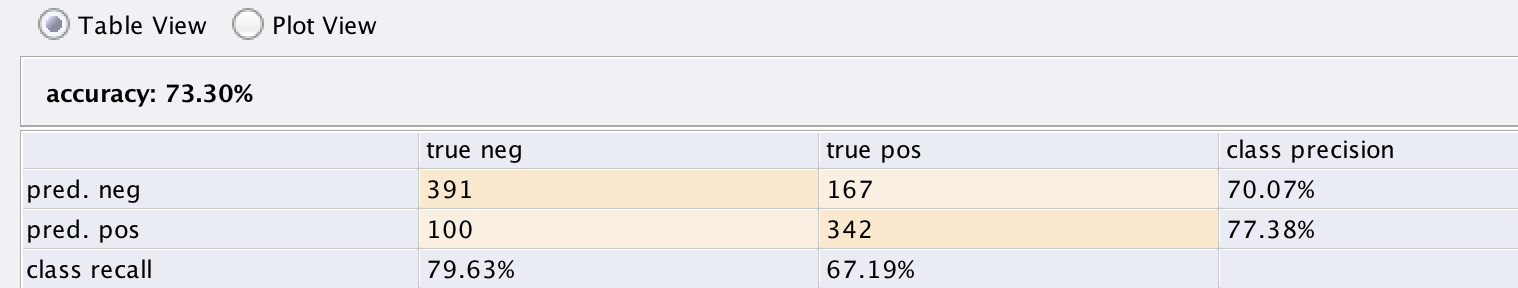
****

**Testing data**

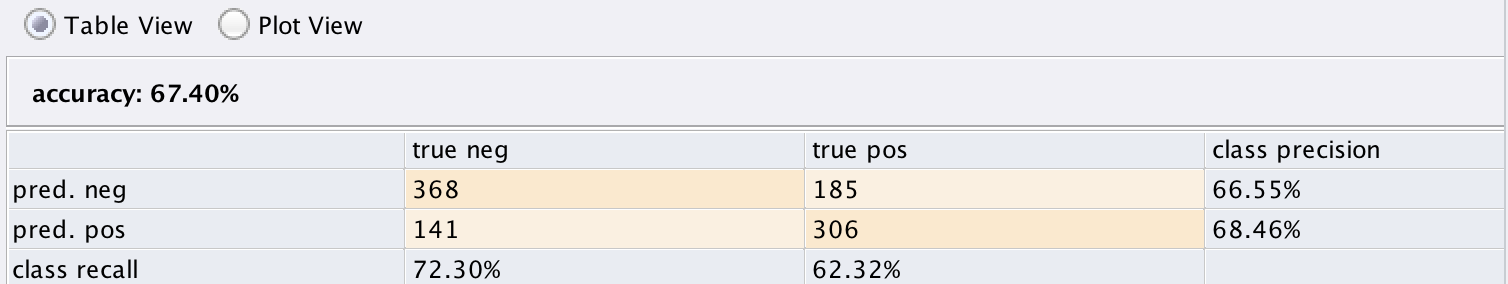
****

**4.2.5 K = 30**

**Training data**

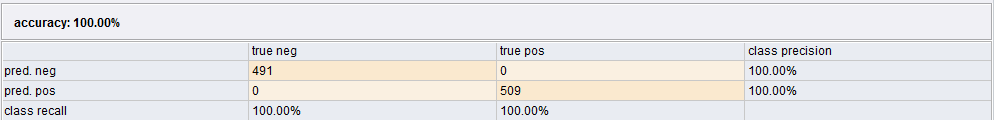
****

**Testing data**

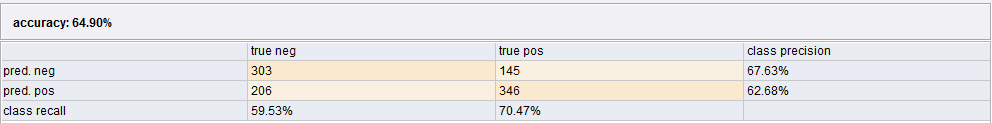
****

**4.2.6 Naïve Bayes**

**Training data**

****

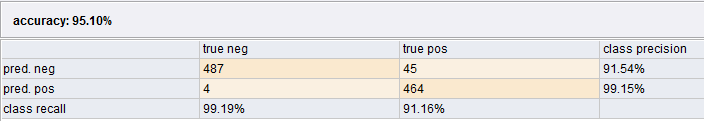
**Testing data**

****

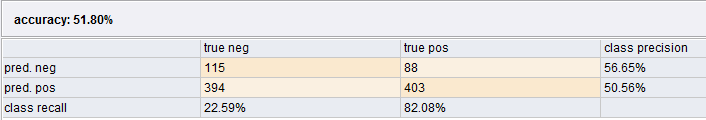
**4.3 Measures defining the document term matrix** [**[Back to the top]**](#t32)

**4.3.1 Binary term occurrence**

**Training data**

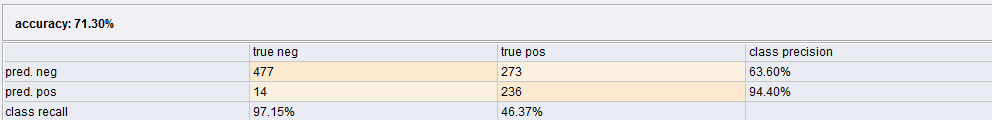
****

**Testing data**

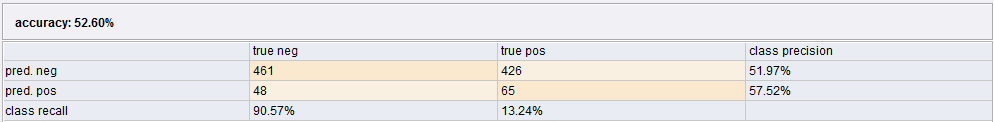
****

**4.3.2 Term occurrence**

**Training data**

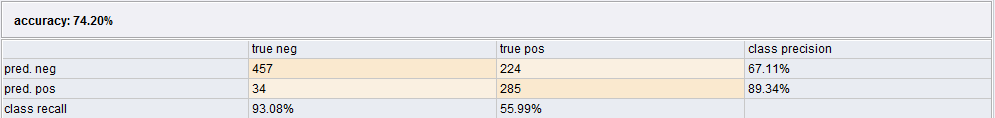
****

**Testing data**

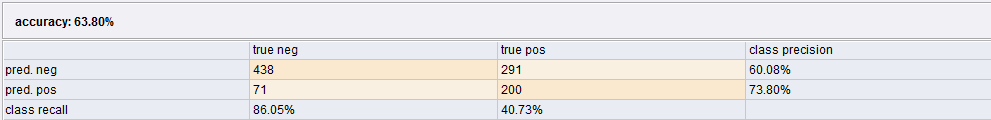
****

**4.3.3 Term frequency**

**Training data**

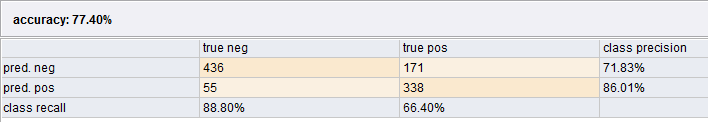
****

**Testing data**

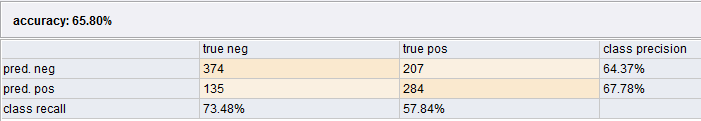
****

**4.3.4 TF-IDF**

**Training data**

****

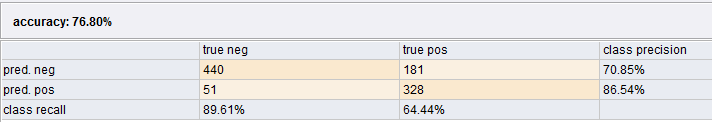
**Testing data**

****

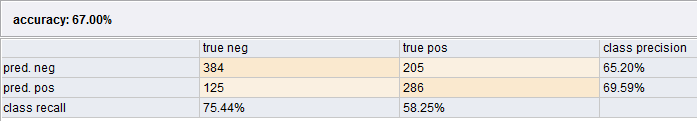
**4.4 Without and with stemming** [**[Back to the top]**](#t33)

**4.4.1 With stemming**

**Training data**

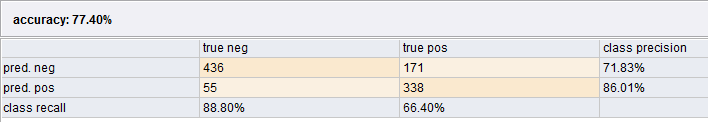
****

**Testing data**

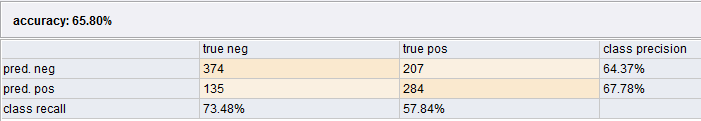
****

**4.4.2 Without stemming**

**Training data**

****

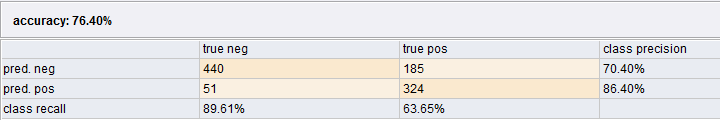
**Testing data**

****

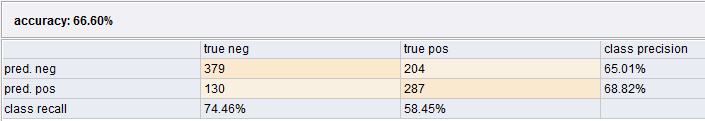
**4.5 POS tags** [**[Back to the top]**](#t34)

**4.5.1 POS = JJ.\*|NN.\*|RB.\*|VB.\***

**Training data**

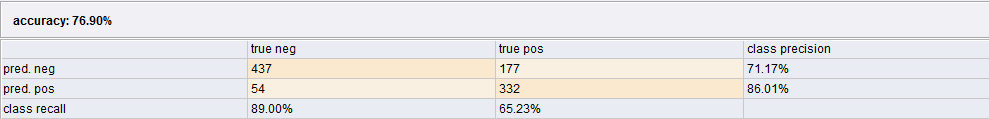
****

**Testing data**

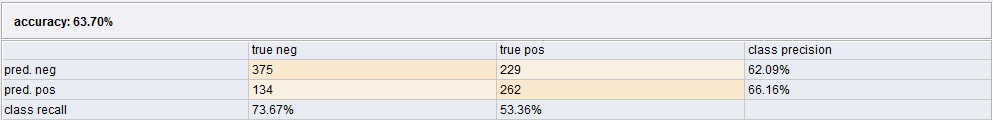
****

**4.5.2 POS = NN.\*|RB.\*|VB.\***

**Training data:**

****

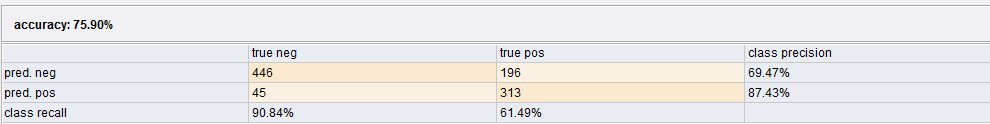
**Testing data:**

****

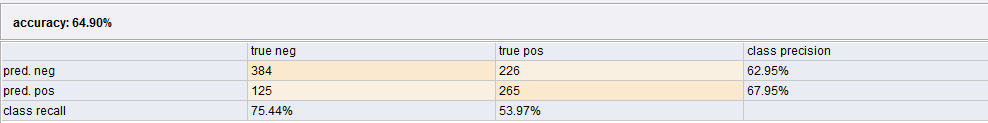
**4.6 Pruning** [**[Back to the top]**](#t36)

**4.6.1 With pruning**

**Training data:**

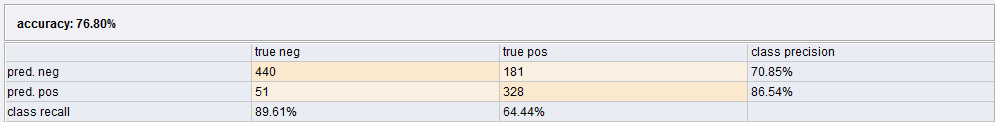
****

**Testing data:**

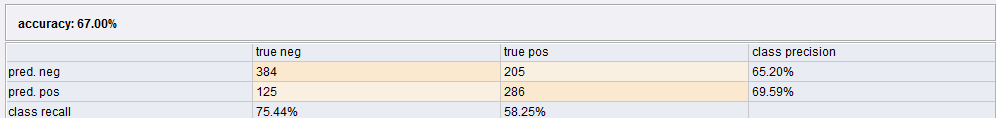
****

**4.6.2 Without pruning**

**Training data:**

****

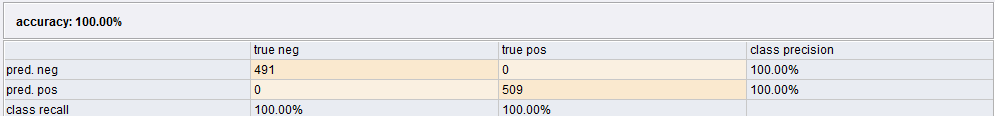
**Testing data:**

****

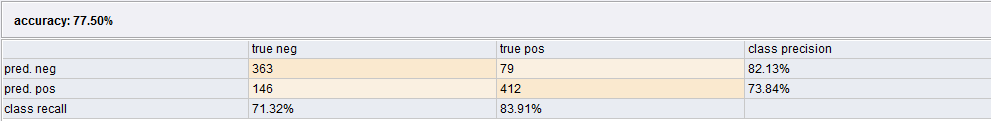
**4.7Comparison of different models** [**[Back to the top]**](#t37)

**4.7.1 SVM with dot-kernel**

**Training data:**

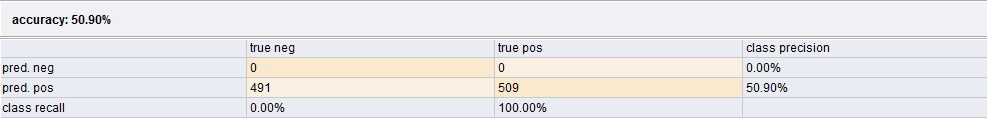
****

**Testing data:**

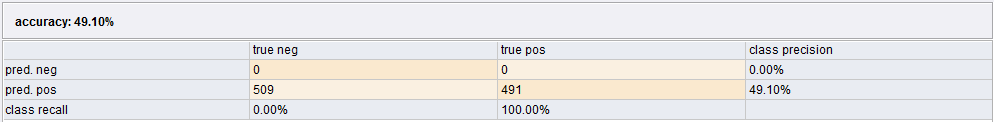
****

**4.7.2 SVM with radial-kernel**

**Training data:**

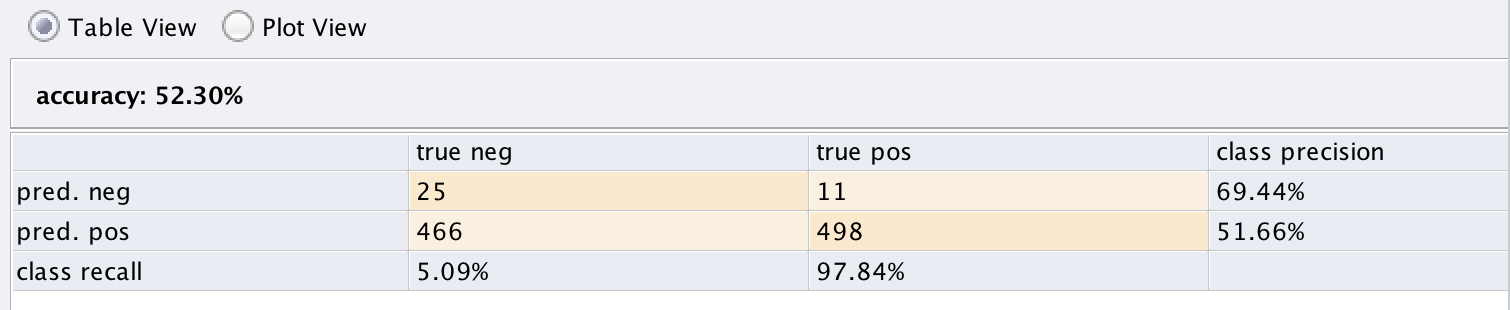
****

**Testing data:**

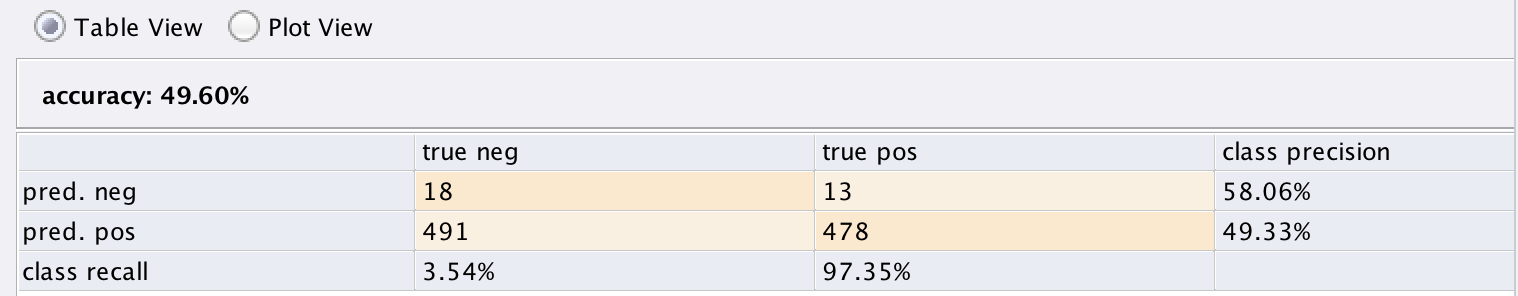
****

4.7.3 Random forest

Training data

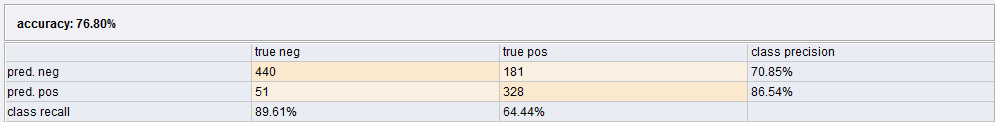


Testing data

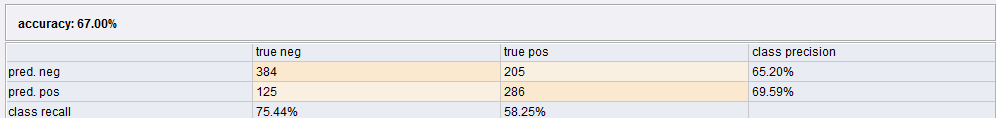


4.7.4 KNN with K = 4

**Training data:**

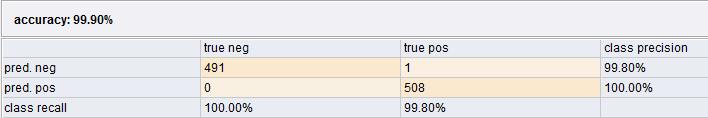
****

**Testing data:**

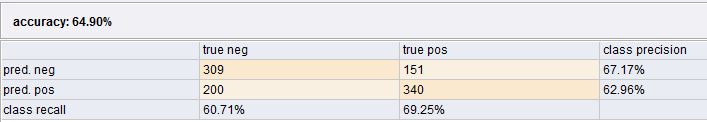
****

4.7.5 Naïve Bayes

Training data



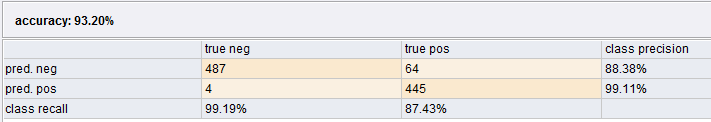
Testing data



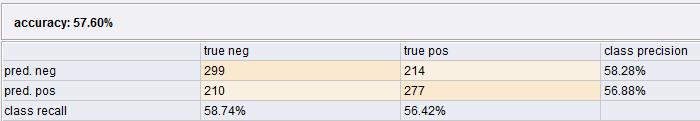
4.8 Comparison of different models using Harvard dictionary [[Back to the top]](#t38)

4.8.1 Naïve Bayes

Training data

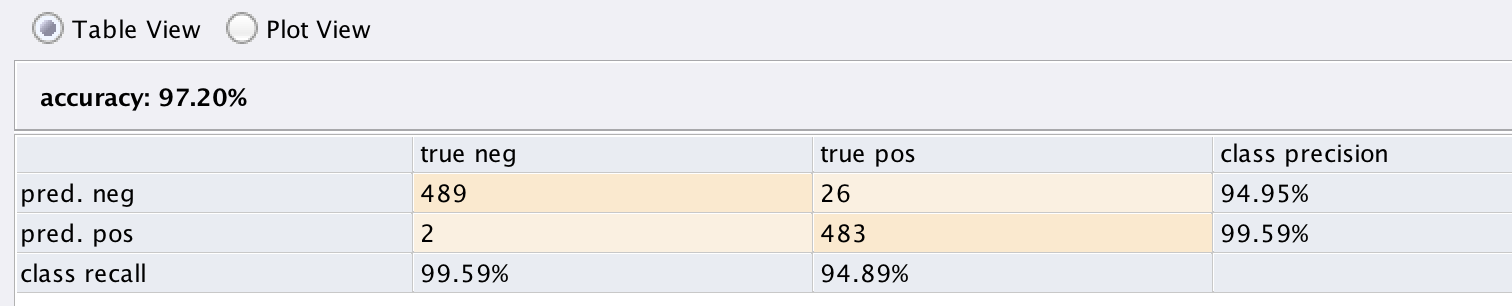


Testing data

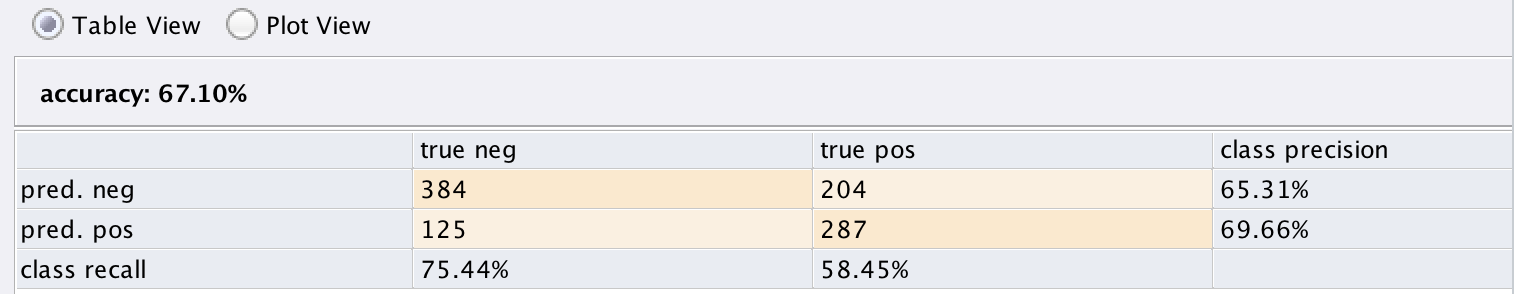


4.8.2 SVM with dot-kernel

Training data

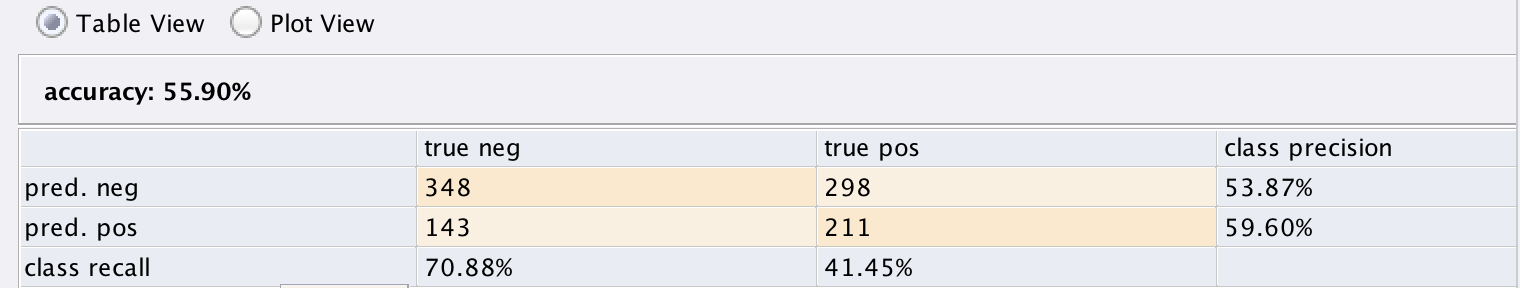


Testing data

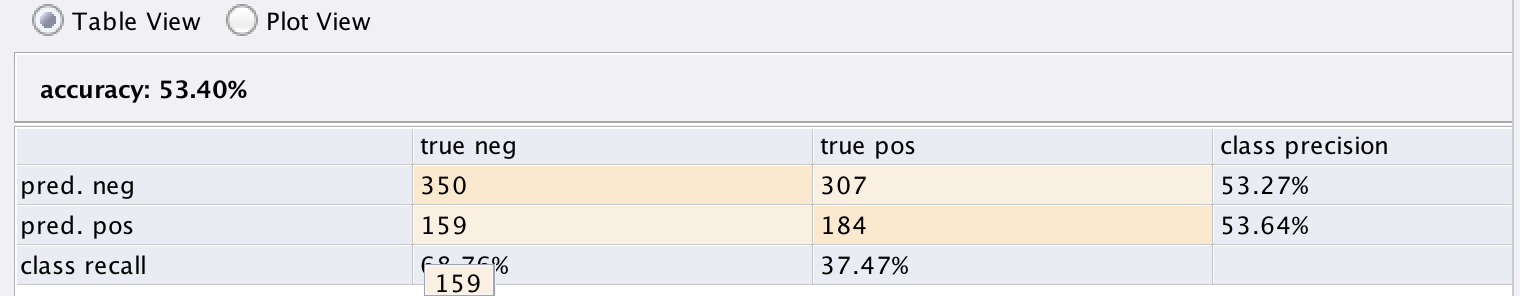


4.8.3 Random forest

Training data

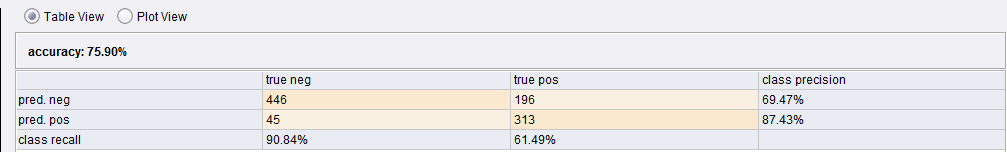


Testing data

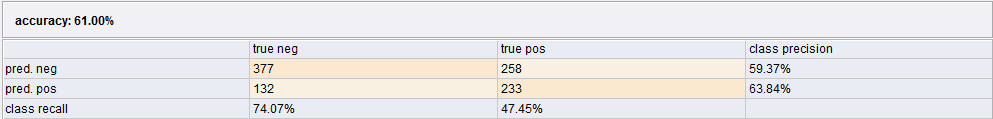


4.8.4 KNN with K=4:

Training data



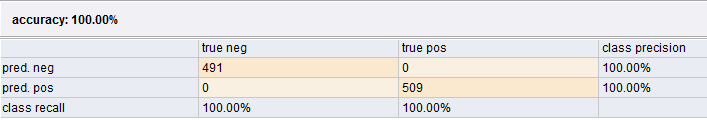
Testing data



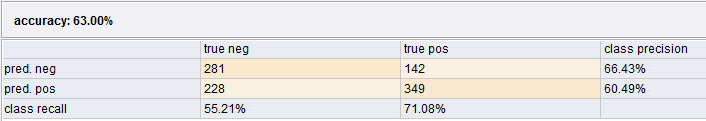
4.9 Word net dictionary [[Back to the top]](#t39)

4.9.1 Naïve bayes

Training data

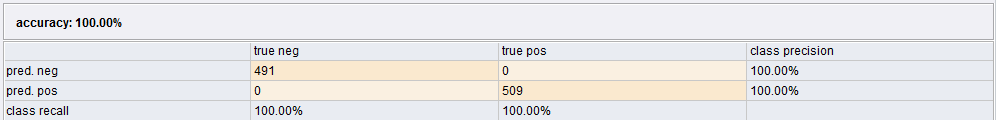


Testing data

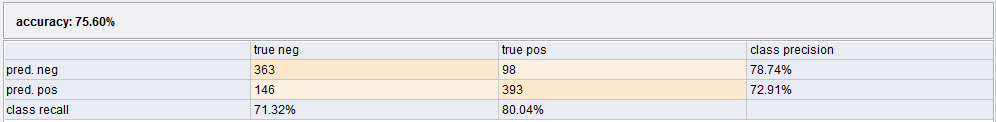


4.9.2 SVM with dot-kernel

Training data

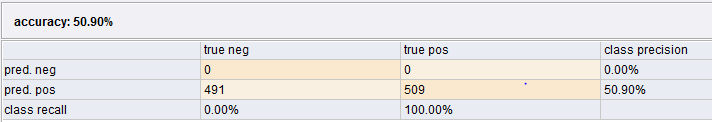


Testing data

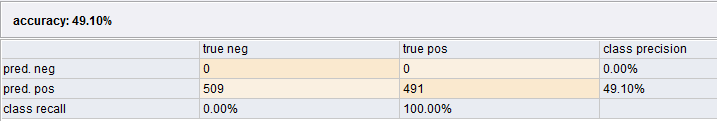


4.9.3 Random forest

Training data

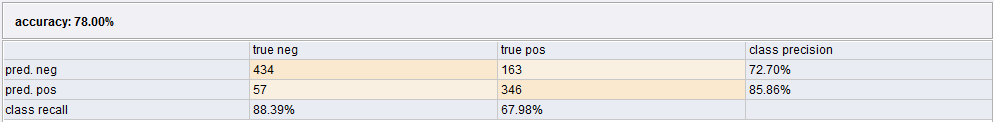


Testing data



4.9.4 KNN with K = 4

Training data



Testing data

