University of New Orleans
Department of Computer Science

Spr 2019: CSCI 6990
Programming Assignment 2

Machine Learning - I

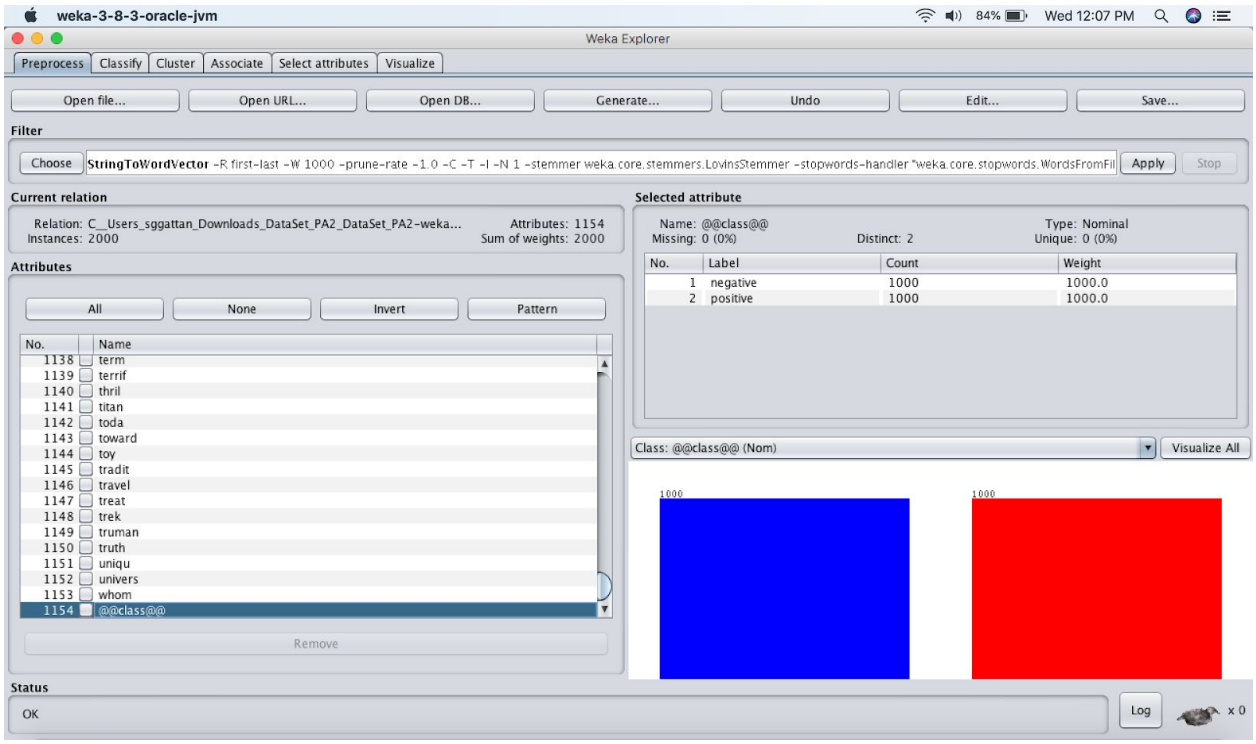
Submitted to:
Professor Dr. Tamjidul Hoque

By
Name: **Astha Sharma**
Student ID: **2564173**

(i)

Using the Weka CLI, a .arrf file was generated from the given positive and negative folders of datasets.

Then the initial class distribution was found, which was:1000 1000 {positive, negative}



The time required conversion time in this step was: 1.37 seconds

(ii)
The string text was converted into the most useful vector using StringToWordVector filtering tool.
This tool converts the string attributes into a set of numeric attributes that represent the word occurrence information from the given text contained in the strings.

The parameters that I have chosen, in particular, are:

- IDFT Transform -

  Role: Sets whether if the word frequencies in a document should be transformed into: fij*log(num of Docs/num of Docs with word i) ; where fij is the frequency of word i in document(instance) j.

  Value set as: TRUE (from FALSE)

- TFT Transform -

  Role: Gets whether if the word frequencies should be transformed into log(1+fij) where fij is the frequency of word i in the document(instance) j.

  Value set as: TRUE (from FALSE)

- Normalizedoclength -

  Role: Gets whether if the word frequencies for a document (instance) should be normalized or not.

  Value set as: Normalize All Data

- Stemmer -

  Role: The stemming algorithm (classname plus parameters) to use.

  Value set as: Lovin Stemmer

Words Collected in this step: 1153

(iii)
The supervised filter ‘infoGainAttributeEval’ with ‘Ranker’ having the threshold set to 0 was applied.

Then the total number of remaining words for classification reduced to 184

The first 10 words with their corresponding information gain value are as given below:

Rank	Information Gain	Words
1	0.07906	bad
2	0.05125	wast
3	0.04998	worst
4	0.04243	stupid
5	0.03562	bor
6	0.03195	suppos

7	0.02968	t
8	0.02655	poor
9	0.02638	perfect
10	0.02503	ridicl

(iv)
10 different classifiers were run one at a time using 10 FCV. The performance and confusion matrix obtained by running the 10 different classifiers are as given below:

- **Naive Baye's Classifier:**

  Accuracy: 81.75 %

  === Confusion Matrix ===

  a b <-- classified as
  811 189 | a = negative
  176 824 | b = positive

- **Logistic Classifier:**

  Accuracy: 84.65 %

  === Confusion Matrix ===

  a b <-- classified as
  848 152 | a = negative
  155 845 | b = positive

- **Multilayer Perceptron:**

  Accuracy: 84.5 %

  === Confusion Matrix ===

  a b <-- classified as
  861 139 | a = negative
  171 829 | b = positive

- **SMO: RBF kernel**

  Accuracy: 85.15 %

  === Confusion Matrix ===

a b <-- classified as
838 162 | a = negative
135 865 | b = positive

- **KNN**

Accuracy: 75.55 %
=== Confusion Matrix ===

a b <-- classified as
598 402 | a = negative
87 913 | b = positive

- **Random Forest (depth:50)**

Accuracy: 82.1 %

=== Confusion Matrix ===

a b <-- classified as
825 175 | a = negative
183 817 | b = positive

- **J48**

Accuracy: 69.2 %

=== Confusion Matrix ===

a b <-- classified as
708 292 | a = negative
324 676 | b = positive

- **AdaBooster - Classifier: SMO, RBF Kernel**

Accuracy: 85.5 %

=== Confusion Matrix ===

a b <-- classified as
844 156 | a = negative
134 866 | b = positive

- **LWL**

Accuracy: 61.6 %

=== Confusion Matrix ===

a b <-- classified as
395 605 | a = negative

163 837 | b = positive

- **SGD**

Accuracy: 84.25 %

=== Confusion Matrix ===

a b <-- classified as
831 169 | a = negative
146 854 | b = positive

(v)
The best method I've found is the **AdaBooster - Classifier: SMO, RBF Kernel.** Its parameters are as given below:

batchSize: 100
Classifier: SMO (Support Vector Machine), RBF Kernel
Debug: False
doNotCheckCapabilities: False
numDecimalPlaces: 2
numIterations: 10
Seed: 1
useResampling: False
weightThreshold: 100

Classifier Output:

Weight: 0.24

Number of performed Iterations: 10

Time taken to build model: 216.8 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	1710	85.5 %
Incorrectly Classified Instances	290	14.5 %
Kappa statistic	0.71	
Mean absolute error	0.1835	
Root mean squared error	0.3464	
Relative absolute error	36.7063 %	

Root relative squared error 69.2714 %
Total Number of Instances 2000

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.844	0.134	0.863	0.844	0.853	0.710	0.906	0.896	negative
	0.866	0.156	0.847	0.866	0.857	0.710	0.906	0.893	positive
Wt Avg.	0.855	0.145	0.855	0.855	0.855	0.710	0.906	0.895	

=== Confusion Matrix ===

a	b	<-- classified as
844	156	a = negative
134	866	b = positive

This method is the best of all the other 9 classification methods as it is giving the utmost performance with the highest accuracy of **85.5%**

Also, the number of correct and incorrect predictions from the confusion matrix is satisfactory.

(vi)

'infoGainAttributeEval':

It is a very popular feature selection technique that is used to evaluate attributes individually[2] or, in particular, calculate the information gain[1]. The entropy or information gain for each attribute for the output variable can be calculated with this technique. The value of the information gain varies from 0(min) to 1(max). The attribute with a higher value for information gain is supposed to contribute more information and therefore, can be selected. On the other hand, the attributes with a lower value of information gain might not add value, so they can be discarded.

When using InfoGainAttributeEval as an evaluator, a search method called Ranker is selected in order to rank all the attributes regarding the evaluation results[2]. This 'Ranker' method treats missing values as a separate value for the attributes.

Running this technique on our dataset, we can see that one attribute contribute more information than the other. We can set an arbitrary cutoff value to discard all of the data that have less contribution.

References:

1. Witten, Ian & Hall, Mark & Frank, Eibe & Holmes, Geoffrey & Pfahringer, Bernhard & Reutemann, Peter. (2009). The WEKA data mining software: An update. SIGKDD Explorations. 11. 10-18. 10.1145/1656274.1656278.
2. https://machinelearningmastery.com/perform-feature-selection-machine-learning-data-weka/
3. https://storm.cis.fordham.edu/~yli/documents/CISC4631Spring17/Preprocess.pdf