Ted Holmberg

Spring 2019: CSCI 6990: Programming Assignment #2:

a 'Movie Review' Predictor in Weka

Abstract:

In this project, Weka is used to generate a predictor model for a movie review dataset. The model must predict based on the text contents whether it is negative or positive.

Table of Contents:

Problem Statement:

Training Dataset: Movie review dataset has been collected for sentiment analysis (see http://www.cs.cornell.edu/people/pabo/movie-review-data/). The dataset has been grouped into positive and negation classes (check Moodle for dataset).

Task [Marks 100]: Develop the **analysis report** as described below and submit it: As demonstrated and discussed, the development of NASA's patent-classifier for fifteen-class classification problem, here similarly for the assignments, we will need to do the following steps and develop the **analysis report**:

i) [**10** points] Given the dataset, use Weka's Simple-CLI to build up the initial ARFF file, which will contain the movie-review-text as a string and the output class {positive, negative}. Add the initial class distribution in the report. Also, report the required conversion time in this step.

ii) [**15** points] Convert the text-string to most useful vector using Weka's unsupervised filtering tool: StringToWordVector. Report the parameters you have chosen and explain their roles and justify your selections. Also, report how many words you have collected in this step.

iii) [**15** points] Using Weka's supervised filter, apply '**infoGainAttributeEval**' with '**Ranker'** having threshold value set to 0.0. Now, report the total words remains for classification and report the first 10 words with their information-gain values.

iv) [**20** points] Run 10 different classifiers and measure their performances using 10 FCV. Report all their performances (accuracy in %) including the confusion matrices. You must include Naïve-Bayes approach as one of the 10 classifiers.

v) [**20** points] Report the best method with its parameter(s) you have found including the performance-evaluation matrices. Explain, why do you think your selected top method is the best method out of the 10 methods you tried.

vi) [**20** points] Review literature to explain '**infoGainAttributeEval**' in details and cite the relevant reference(s). Submit the copies of the paper(s) that you have cited to explain '**infoGainAttributeEval**'.

i) [**10** points] Given the dataset, use Weka's Simple-CLI to build up the initial ARFF file, which will contain the movie-review-text as a string and the output class {positive, negative}. Add the initial class distribution in the report. Also, report the required conversion time in this step.

STEP 1: DATA IMPORTATION INTO WEKA:

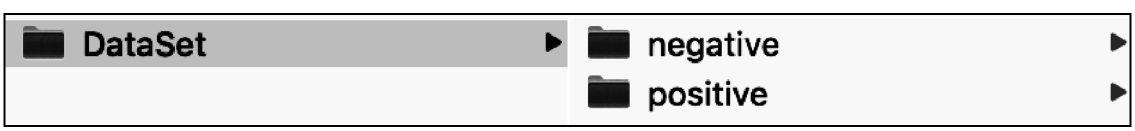
The given '*Movie Review*' dataset is contained across two separate directories (labeled: positive, negative). The directory labeled *positive* contains 1000 text documents (.txt), where each file is a positive movie review. The directory labeled as *negative* contains 1000 text files (.txt), where each file is a negative movie review.

This dataset must be preprocessed and reformatted before any analysis may begin. Since Weka is the tools selected analysis and classification, then the dataset must be converted into an ARFF format. ARFF stands for Attribute-Relation File Format. It is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka machine learning software.

Weka provides the necessary converters to reformat the 'Movie Review' dataset to ARFF. As Weka is a Java-based application, its data conversion tools may execute from the command line via calls to the Weka.jar file, package: weka.core.converters, class: TextDirectoryLoader. This is optimal, as it allows data preprocessing into the ARFF format to be automated via system-level scripts. For the purposes of this project, the dataset conversion was accomplished using a python script. See *Appendix 1: Weka Preprocess Data - Time Capture.*

The Weka TextDirectoryLoader requires the filepath for the dataset. For this project, the dataset filepath should be the parent directory of the two subdirectories: negative and positive.

Path to DataSet



Weka will then use the directory labels to produce a classifier for the text files within those directories. Thus the text files within the negative directory are labeled negative and the text files within the positive directory are labeled with as positive in the resulting ARFF file.

The runtime to convert the original dataset of 2000 text files into ARFF format took precisely **4790.44116211 milliseconds** or **~4.8 seconds**. This time calculation was achieved by executing the weka jar from a bash command via a python script. The python script captured a before and after timestamps to determine the total runtime. See *Appendix 1: Weka Preprocess Data - Time Capture* for the python implementation. Instructions for running this python script is included within the implementation.

After converting the dataset into ARFF we can examine the contents using Weka as a viewer. The converted ARFF dataset initially has three columns, row number, text content, anda positive/negative label.

Movie Review' Dataset (via Weka ARFF Viewer)

ARFF-Viewer - /Users/ted/Desktop/Machine-Learning/MachineLearning1-HW3/DataSet_PA2/ARFF/ProcessedData.arff

File Edit View

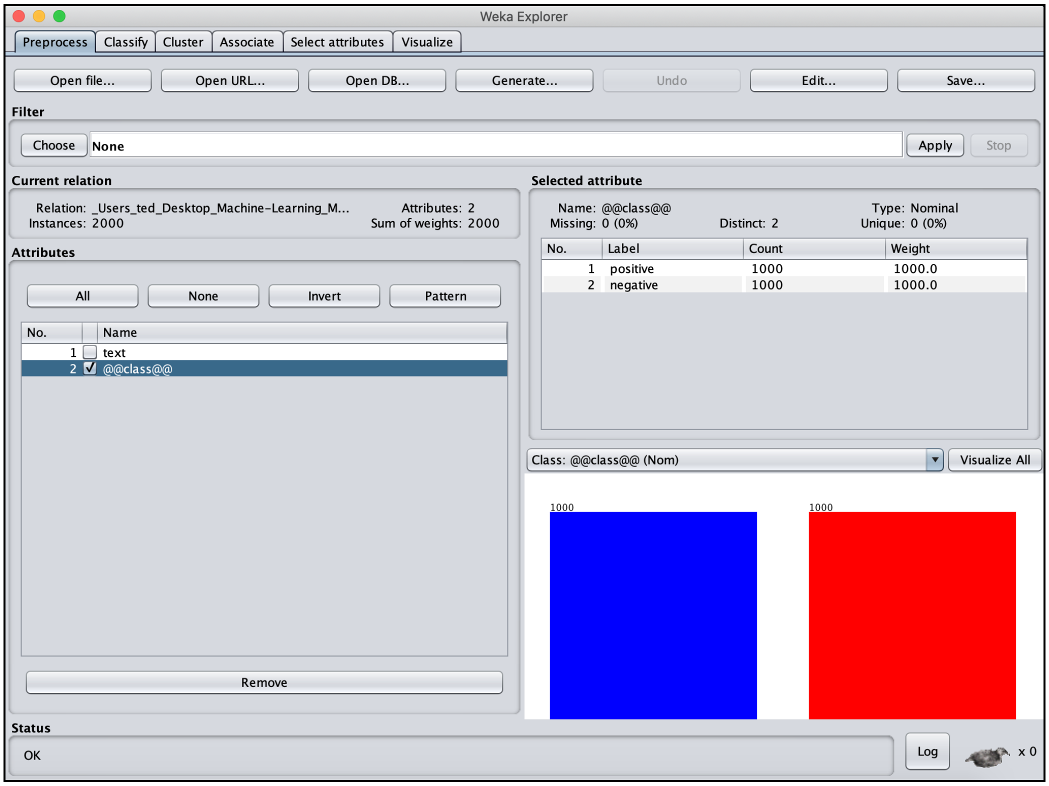
ProcessedData.arff *

Relation: _Users_ted_Desktop_Machine-Learning_MachineLearning1-HW3_DataSet_PA2

No.	1: text String	2: @@class Nominal
1	assume nothing . \nthe phrase is perhaps one of the most used of the 1990's , as first impressions and rumors are hardly ever what they seem to be .	positive
2	@relation _Users_ted_Desktop_Machine-Learning_MachineLearning1-HW3_DataSet_PA2\n\n@attribute text string\n@attribute @@class@@	ARFF
3	bad . bad . \nbad . \nthat one word seems to pretty much sums up beyond the valley of the dolls . \nif that summary isn't enough for you , how about	negative
4	plot : derek zoolander is a male model . \nhe is also very dumb and impressionable . \nfor that reason , he is secretly hired and trained (so secret ,	positive
5	isn't it the ultimate sign of a movie's cinematic ineptitude when you can't think of much to say about it other than " it sucks " ? \none of the first official year	negative
6	i actually am a fan of the original 1961 or so live-action-disney flick of the same name starring hayley mills twice as a pair of twins , separated at birth by	positive
7	" gordy " is not a movie , it is a 90-minute-long " sesame street " skit , and a very bad one at that . \nthis movie is so stupid and dumb that it's	negative
8	a movie that's been as highly built up as the truman show , with reviews boasting , " the film of the decade ! " \nand " a breakthrough ! " \ncan only be	positive
9	disconnect the phone line . \ndon't accept the charges . \ndo anything you can to avoid the wretched , melodramatic sisterhood dramedy `hanging up' .	negative
10	" good will hunting " is two movies in one : an independent take on the struggle of four boston pals and a traditional hollywood , " prodigy child " film	positive
11	when robert forster found himself famous again after appearing in " jackie brown " , he immediately signed up for a little film called " american perfekt " .	negative
12	the story of us , a rob reiner film , is the second movie this fall that touches the viewer in a way they are rarely touched by a film , as they can see their	positive
13	this is my first review that i post to this newsgroup , and i kind of feel like i have to say something negative about this film . \nno one else seems to care	negative
14	anastasia contains something that has been lacking from all of the recent disney releases . . . \n (especially hercules) . . . \nemotion . \nall the wacky	positive
15	" lake placid " marks yet another entry in the series of " predator pics " that were a screen staple in the late 1970s (post- " jaws ") and were revived	negative
16	" the fighting sullivans " contains a major plot development in the last ten minutes that every movie guide has seen fit to give away . \nthere was no	positive
17	the main problem with martin lawrence's pet project , a thin line between love and hate , like any fatal attraction variation where the protagonist is a man	negative
18	george little (jonathan lipnicki) wants a little brother . \nafter mr . and mrs . little (hugh laurie and geena davis) visit the orphange , they decide to	positive
19	" with all that education , you should know what happiness is . " \nstarring sylvia chang , teresa hu , hsu ming , li lieh , mao hsueh-wei ; directed by	negative
20	before you read my review , you gotta know that i love woody allen . \nthis is a very important note because allen's films are generally an acquired taste	positive
21	jet li busted onto the american action movie scene , when he stole the show in 1998's lethal weapon 4 , with his wicked looks , his nasty moves and his	negative
22	who would have thought ? \njim carrey does drama . \nwhen i first saw the advertisement for the truman show , i thought , " what a hilarious idea for a	positive
23	starring shawnee smith ; donovan leitch ; ricky paull goldin ; kevin dillon & billy beck the blob is the remake of the 1960's classic (a term that i use very	negative
24	i rented " brokedown palace " last night blind , having heard nothing about it beforehand , and i enjoyed it immensely despite some flaws . \nfor anyone	positive
25	in 1970s , many european intellectuals , especially those on the left political hemisphere , became obsessed with the rise of fascism . \nwhich wasn't so	negative
26	the thought-provoking question of tradition over morals is the subject directly at the core of " leila , " a powerfully articulated and subtle drama from	positive
27	the army comedy genre has never turned out a truly good movie (if you don't count neil simon's biloxi blues) . \nyear after year , more predictably	negative
28	first , i am not a big fan of the x-files tv series . \ni have nothing against it particularly , i just don't happen to watch it . \nhaving said that , i can now say	positive
29	and just when you thought joblo was getting a little soft around the corners , not rating anything lower than your standard " this movie sucks " , along	negative
30	synopsis : committed to an asylum , the marquis de sade (rush) continues to publish pornographic literature , aided by young maid madeleine (winslet)	positive
31	talk about beating a dead horse ! \nwhen home alone was released in 1990 , it was a breath of fresh air , and the final box office tally indicated how	negative
32	plot : a group of asbestos cleaners get a job removing the gunk from an old insane asylum . \nas each day passes , the crew members begin to discover	positive
33	capsule : godawful " comedy " that's amazingly shabby and cut-rate , and rather bereft of laughs . \ni was having a bad week in my life when i saw austin	negative
34	[note that followups are directed to rec . arts . movies . current-films and rec . arts . movies . startrek . current \nonly , not to rec . arts . sf . movies .	positive
35	in our time . \nin our modern world , where the cool rule , it's hard to imagine that shakespeare is becoming 'the man' . \nand yet - film after film , after	negative
36	meet joe black (reviewed on nov . 27/98) \nstarring brad pitt , anthony hopkins , claire forlani \nin " meet joe black " , brad pitt plays death . \nthat's	positive
37	about an hour or so into " the jackal , " a character wandered around as people were being shot at in a big suspense sequence , and one of the audience	negative
38	i had lost all faith in pg-13 movies that are intended for teenagers and adults . \nthe last dozen or so that i have seen have all felt incomplete , as if the	positive
39	i'm not sure who the genius is who came up with the idea of comparing " disturbing behavior " with " scream . " \nmaybe it's because they're both horror	negative

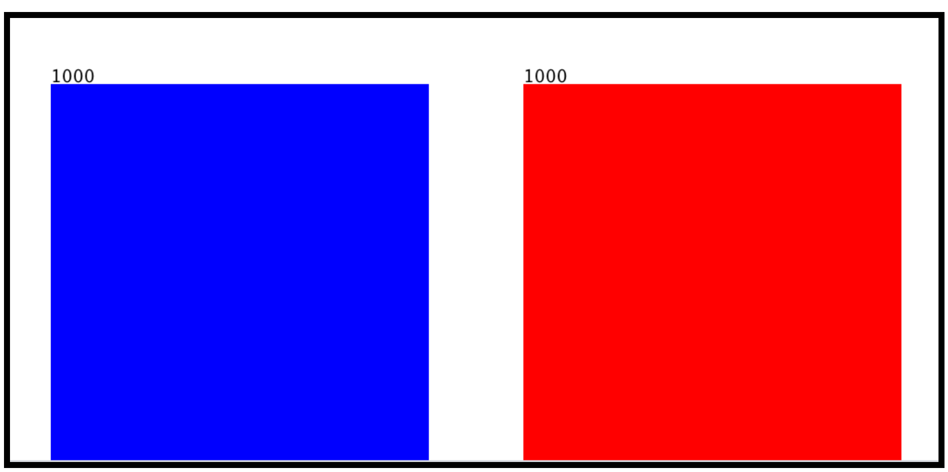
Loading the 'Movie Review' dataset into Weka. When Weka is launched, it offers the option to open file. Since ARFF is supported by Weka, it preloads information regarding the contents of the dataset such as the initial class distributions in both tabular and graphical formats.

Weka Explorer for the 'Movie Review' Dataset



The class distribution of the 'Movie Review' dataset when the ARFF is initially loaded into Weka amy be graphically displayed.

Initial Class Distribution



No.	Label	Count	Weight
1	positive	1000	1000.0
2	negative	1000	1000.0

ii) [**15** points] Convert the text-string to most useful vector using Weka's unsupervised filtering tool: *StringToWordVector*. Report the parameters you have chosen and explain their roles and justify your selections. Also, report how many words you have collected in this step.

STEP 2: DATA PREPARATION FOR WEKA (TOKENIZING):

Cleaning & Tokenization: The 'movie review' dataset is now imported into Weka, however, it must be optimized before any real analysis may begin. Currently, the input feature data for each row is expressed as a single String containing the full text contents of each review. This is not a practical format for performing any analysis, classifying, or clustering actions. This should instead be converted into a more efficient data structure in the form of a Word Vector. To do this, we must tokenize the text data such that it may be vectorized.

What is Tokenization: To make the provided text document classifiable using Machine Learning we need to do feature extraction that is converting the normal text to a set of features that can then be used by the ML Algorithm to discriminate between negative and positive reviews.

Weka provides built-in preprocessing filters explicitly for this purpose, i.e. converting text data into vector types. According to the Weka documentation, the *StringToWordVector* method performs the following actions with additional options. See *Appendix 2: StringToWordVector API Documentation*

public class StringToWordVector
Converts string attributes into a set of numeric attributes representing word occurrence information from the text contained in the strings. The dictionary is determined from the first batch of data filtered (typically training data). Note that this filter is not strictly unsupervised when a class attribute is set because it creates a separate dictionary for each class and then merges them.

Options Name	**Description**
attributeNamePrefix	Prefix for the created attribute names. (default: "")
stopwordsHandler	The stopwords handler to use (Null means no stopwords are used).
wordsToKeep	The number of words (per class if there is a class attribute assigned) to attempt to keep.

debug	If set to true, filter may output additional info to the console.
outputWordCounts	Output word counts rather than boolean 0 or 1(indicating presence or absence of a word).
lowerCaseTokens	If set then all the word tokens are converted to lowercase before being added to the dictionary.
tokenizer	The tokenizing algorithm to use on the strings.
doNotCheckCapabilities	If set, the filter's capabilities are not checked before it is built. (Use with caution to reduce runtime.)
doNotOperateOnPerClassBasis	If this is set, the maximum number of words and the minimum term frequency is not enforced on a per-class basis but based on the documents in all the classes (even if a class attribute is set).
attributeIndices	Specify range of attributes to act on. This is a comma separated list of attribute indices, with "first" and "last" valid values. Specify an inclusive range with "-". E.g: "first-3,5,6-10,last".
normalizeDocLength	Sets whether if the word frequencies for a document (instance) should be normalized or not.
saveDictionaryInBinaryForm	Save the dictionary as a binary serialized java object instead of in plain text form.
invertSelection	Set attribute selection mode. If false, only selected attributes in the range will be worked on; if true, only non-selected attributes will be processed.
minTermFreq	Sets the minimum term frequency. This is enforced on a per-class basis.
TFTransform	Sets whether if the word frequencies should be transformed into log(1+fij) where fij is the frequency of word i in document (instance) j.
periodicPruning -	Specify the rate (x% of the input dataset) at which to periodically prune the dictionary. wordsToKeep prunes after creating a full dictionary. You may not have enough memory for this approach.
stemmer	The stemming algorithm to use on the words.
dictionaryFileToSaveTo	The path to save the dictionary file to - an empty path or a path '-- set me --' means do not save the dictionary.
IDFTransform	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

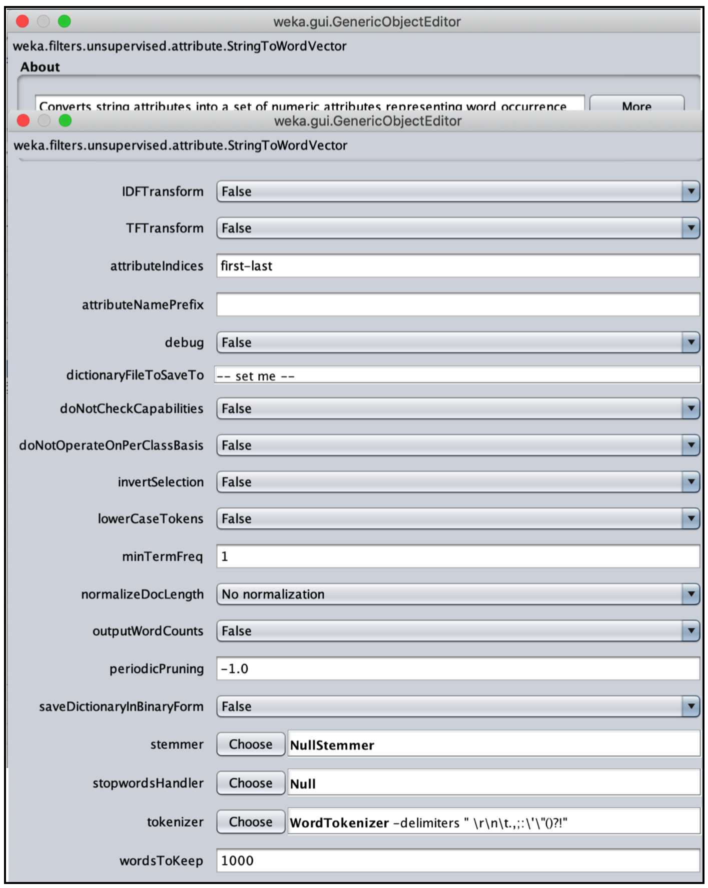
To select the *StringToWordVector* filter from in Weka:

1. Click on *Choose* button below Filter
2. Choose: *weka → filters → unsupervised → attribute → StringToWordVector*

Weka GUI with Filter field (with StringToWordVector selected)

Options: The StringToWordVector filter has several different options. The default values are shown below. However, adjusting the options can improve the tokenization of the text into features.

Weka GUI - StringToWordVector Options (Default Settings)

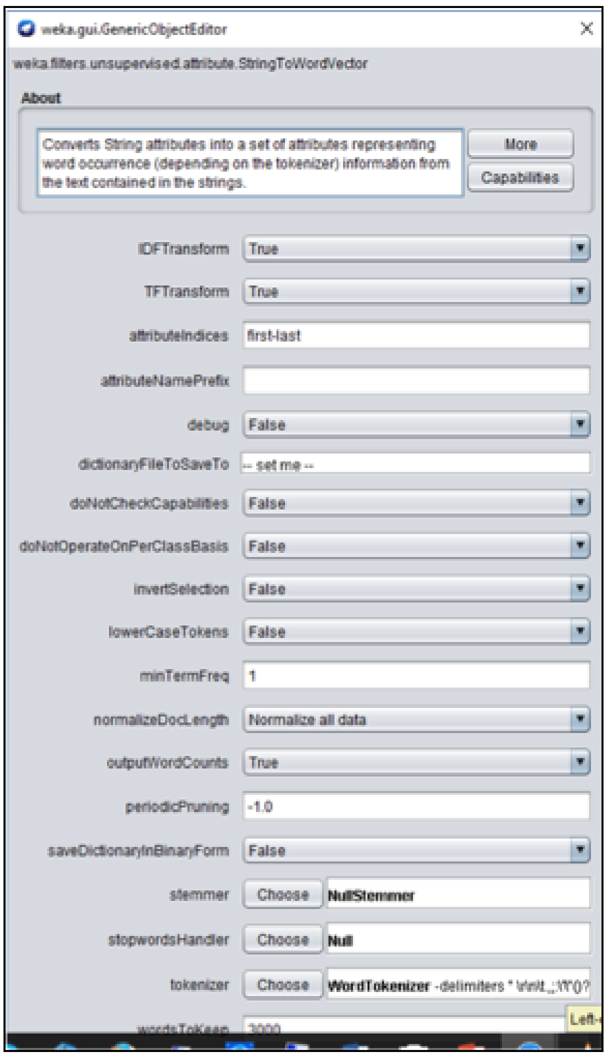


Default Options: The 'movie review' dataset tokenized using the default options from StringToWordVector filter results in identifying the occurrence of 1165 attributes (i.e. singular words). All 1165 attributes have bimodal distributions in occurences.

Custom Options: Adjusting StringToWordVector options with the following updates.

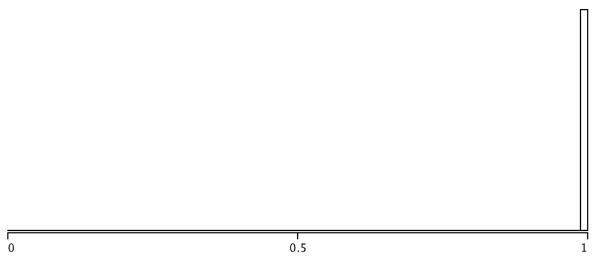
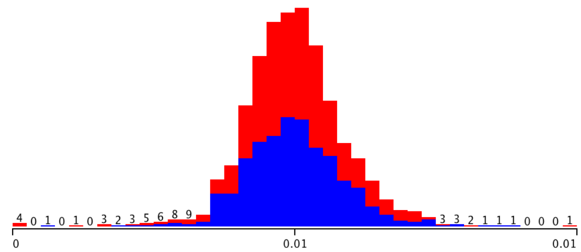
Term Frequency (TF)/Inverse Document Frequency (IDF) options: Models how important a word is to a given document within a collection of documents. It is often used as a weighting factor in searches of information retrieval and text mining. The TF–IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the total collection that contain the word, which helps to adjust for the fact that some words appear more frequently in general. TF–IDF is one of the most popular term-weighting schemes today;

Weka GUI - StringToWordVector filter - (Custom Options)



IDFTransform: **True** Reason: Turn on the weighing factor of Inverse Document Frequency. This helps track how important a word is in a given document
TFTTransform: **True** Reason: Turn on the weighing factor of Term Frequency. Captures the frequency of the word appearing.
normalizeDocLength: **Normalize all data** Reason: Normalize all data values between 0-1
outputWordCounts: **True** Reason: Provides greater granularity in word occurence through counts instead of binary: present(1)/absent(0)

The results of this filtering identifies 1165 attributes, however produces better frequency distributions for performing analysis.

*Frequency of: '**a**' (Default Options)*	*Frequency of: '**a**' (Custom Options)*
	

The statistical algorithms must be applied on these attributes to constructo a predictor will perform better with frequencies similar to the right as opposed to the left.

Total number of attributes identified in both cases is **1165 words**.

iii) [**15** points] Using Weka's supervised filter, apply '**infoGainAttributeEval**' with '**Ranker'** having threshold value set to 0.0. Now, report the total words remains for classification and report the first 10 words with their information-gain values.

STEP 3: FEATURE SELECTION AND RANKING:

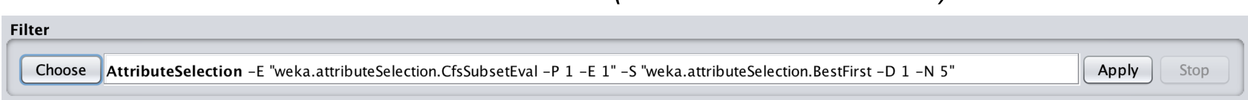
The 'Movie Review' dataset has now been prepared into a set of 1165 input features and one output class: 'positive/negative.' However, there are still too many features to perform a meaningful analysis, so more filtering is needed to generate a predictor that uses only the most critical set of attributes. Constructing better predictor models requires the removal of any words (i.e. features) that do not contribute in determining whether a review is negative or positive. So we must rank the features and select the top features that correlate to the output class. Note: It is important to ensure that your @@*class*@@ attribute defining your positive/negative values is assigned as the output class and appears as the last column in the dataset, as all feature ranking must be compared to the output class.

Weka provides built-in tools for performing feature filtering on datasets within *AttributeSelection* class.

To select the infoGainAttributeEval filter from in Weka:

1. Click on *Choose* button below Filter
2. Choose: *weka → filters → supervised → attribute → AttributeSelection*

Weka GUI - Filter field (with AttributeSelection)



AttributeSelection: Attribute selection requires two components be specified in order to execute. The two parts of Attribute selection are:

- Attribute Evaluator
- Search Method

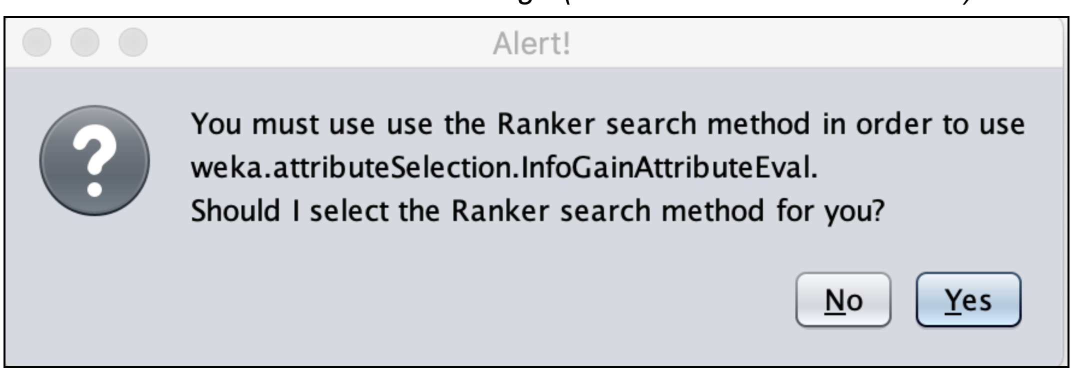
The *attribute evaluator* is the technique by which each attribute in your dataset (also called a column or feature) is evaluated in the context of the output variable (e.g. the class). The *search method* is the technique by which to try or navigate different combinations of attributes in the dataset in order to arrive on a short list of chosen features.

Weka GUI - AttributeSelection (evaluator: InfoGainAttributeEval, search: Ranker)

InfoGainAttributeEval is an evaluator that implements Information Gain, which is an Information Theory metric that takes into consideration how the entropy (or separation) of the space points changes when using one attribute. A high score in Information Gain means it is easier to classify the points.

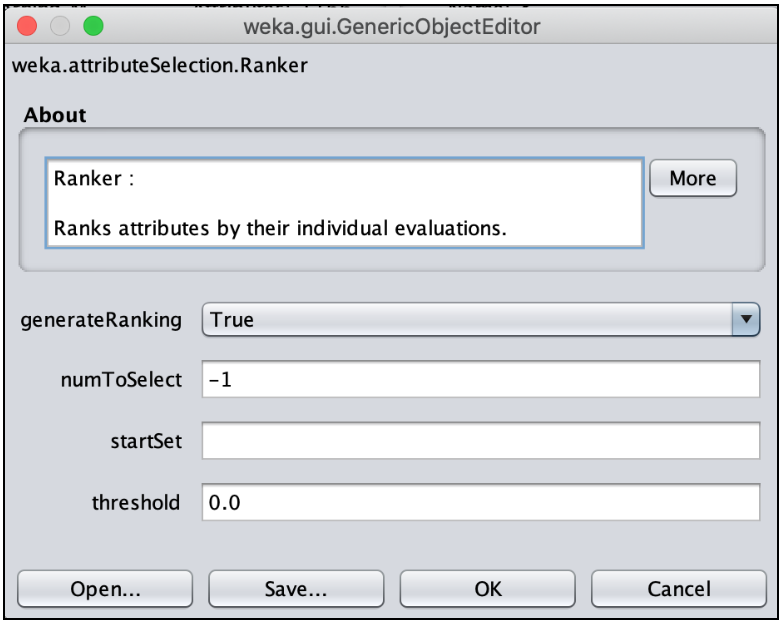
Some Attribute Evaluator techniques require the use of specific Search Methods. For example, the *InfoGainAttributeEval* technique can only be used with a *Ranker* Search Method. When selecting an Attribute Evaluator, the interface may ask you to change the Search Method to something compatible with the chosen technique.

Weka GUI - Alert Message (Evaluator/Search mismatch)



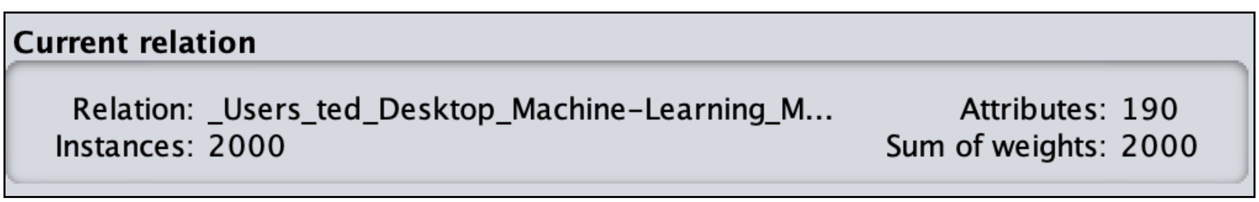
Ranker is a Search Method that evaluates each attribute and lists the results in a rank order. You can assign a threshold for Ranker, which for this project will be set to 0.

Weka GUI - Ranker (threshold: 0)



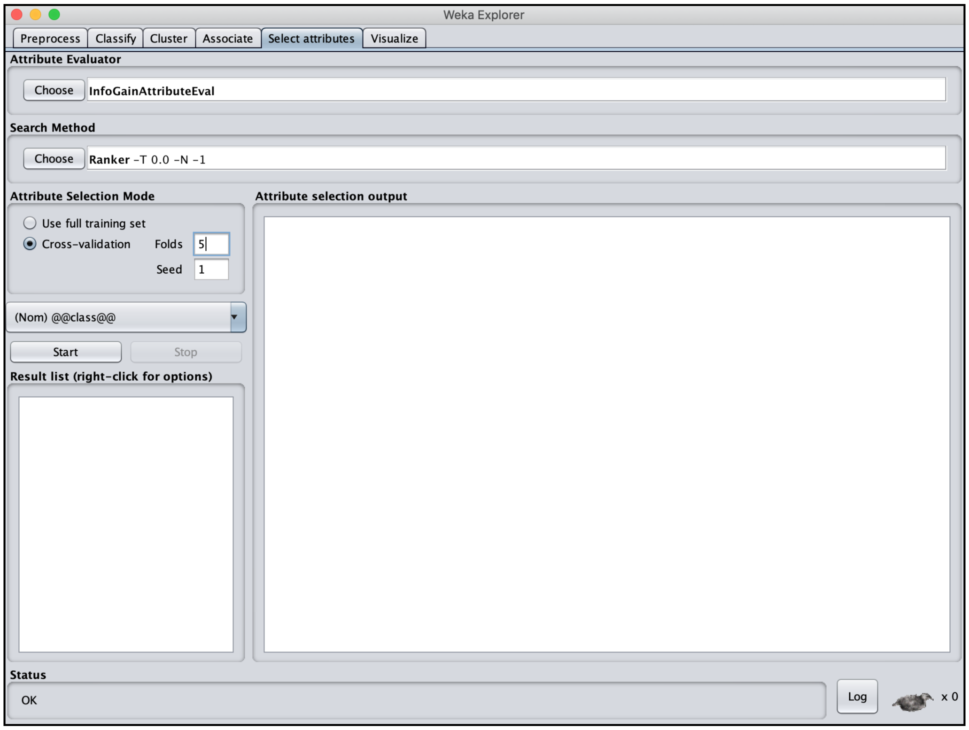
Once the attribute evaluator and search methods are selected, then the AttributeSelection can be executed with the '*Apply*' button and the number of attributes should be filtered down. Reducing the number of attributes lowers the complexity from a higher dimensional space into a lower dimensional space. In this case, the dataset dropped from 1165 independent variables (i.e. dimensions) down to just 190, that's a 83.691% decrease.

Weka GUI Current relation - Attributes: 190



Top Ranked Features:
To get the rankings and scores for these 190 attributes from within Weka, click on the '*Select attribute*' tab at top. Verify that the selected evaluator and search methods are correct. For this project, in the *Attribute Selection Mode* uses Cross-Validation, with 5 folds to perform the attribute selection. Then press the *start* button to execute.

Weka GUI = Select Attributes menu



Weka displays the merits/rankings of the 190 attributes. There are 3 columns of values reported: Average Merit, Average Rank, Attribute

Top 10 Ranked Attributes (by Merit)

```
=== Attribute selection 5 fold cross-validation (stratified), seed: 1 ===

average merit      average rank  attribute
 0.071 +- 0.007       1   +- 0          1 bad
 0.051 +- 0.002       2   +- 0          2 worst
 0.032 +- 0.002       3.6 +- 0.8        4 boring
 0.033 +- 0.004       4   +- 0.89       3 stupid
 0.027 +- 0.003       7   +- 2.28       5 wasted
 0.027 +- 0.002       7.4 +- 1.96       6 t
 0.026 +- 0.001       8   +- 1.41       7 waste
 0.025 +- 0.002       9.6 +- 2.94       8 ridiculous
 0.024 +- 0.003      10.6 +- 3.14      10 supposed
 0.024 +- 0.005      11.6 +- 5.68       9 awful
```

The first column, average merit is the information gain score for that attribute averaged across all folds of the cross validations. The second column, average rank is the ranking of that feature averaged across all folds of the cross validation. The third column, is the attributes index number and the word.

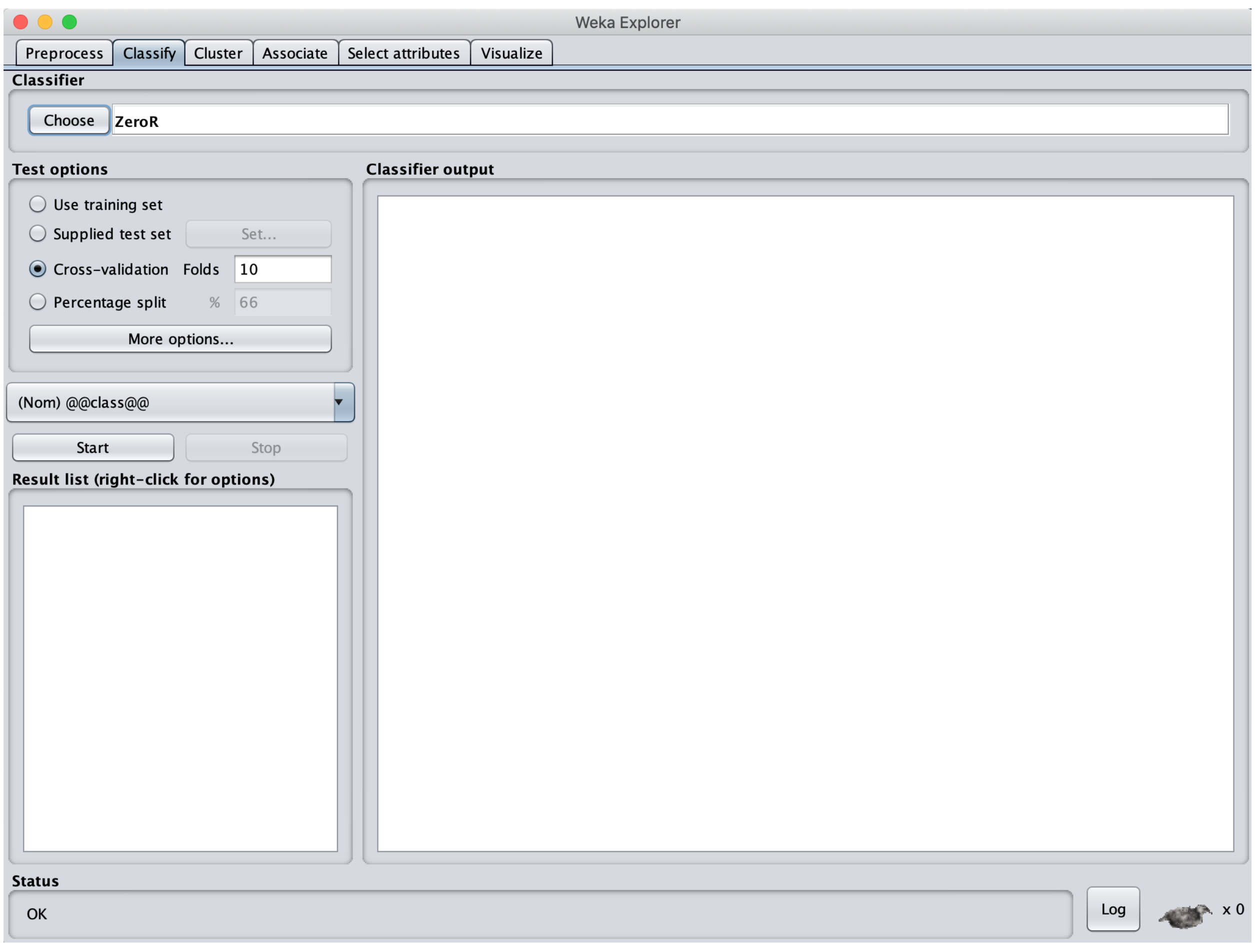
iv) [**20** points] Run 10 different classifiers and measure their performances using 10 FCV. Report all their performances (accuracy in %) including the confusion matrices. You must include Naïve-Bayes approach as one of the 10 classifiers.

STEP 4: MODEL SELECTION

The 'movie review' dataset is completely preprocessed and prepared for training a prediction model. However, there are several ML algorithms that may be used to build a prediction model. Since the type of prediction this model must perform is classification based, i.e. is this review negative or positive, then we may eliminate all models that can't produce a categorical result.

In Weka, the classifier ML algorithms are accessible from the 'Classify' tab at the top. When selected, its possible to choose from a selection of classifiers to build and test a prediction model.

Weka GUI - Classify menu



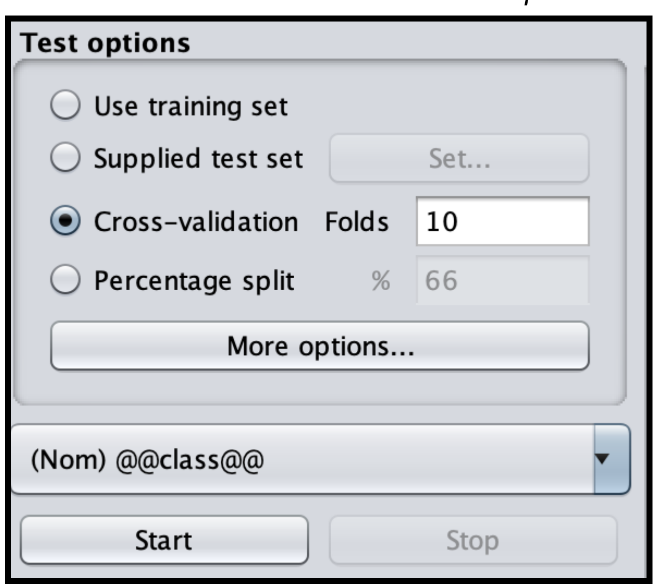
Choosing the Best Classifier:
There a several classifier algorithms that may be chosen to solve construct a predictor. In Weka, there are 48 possible classifier types, each with its own set of optional parameters that may tweak and finetune the predictor's performance.

Unfortunately there is no one ML algorithm that is universally best, it all depends on the particular dataset for a given problem. So the optimal strategy for selecting the best classifier is to try many different approaches with the default settings and then compare them by their prediction accuracy. Then from those initial evaluations, select those models that performed best, and finetune their optional parameters to increase their performance.

I generated 114 models from the 48 possible Weka classifiers. In this report, I have highlighted 13 of those attempts. Most of the attempts below are generated with default options. The most accurate models, I attempted to finetune with custom options. Many of the attempted customizations did not yield improvements and thus are not listed below. In fact, only one classifier, SGD, improved performance during the fine-tuning process. As such, it's the only one that appears multiple attempts.

Testing Model Accuracy:
Cross-validation with 10 folds will be selected to test each model's predictive capabilities. With the output class using the nominal class (positive, negative).

Weka GUI - Classifier Menu - Test Options



CLASSIFICATION ATTEMPT I: k-Nearest Neighbors

weka.classifiers: lazy.IBK, IB1 instance-based classifier

kNN Synopsis:
K-nearest neighbours classifier. Can select appropriate value of K based on cross-validation. Can also do distance weighting.

OPTIONS:
- **kNN: 9**

Attempt I - Results

```
=== Summary ===

Correctly Classified Instances        1460               73      %
Incorrectly Classified Instances       540               27      %
Kappa statistic                          0.46
Mean absolute error                      0.363
Root mean squared error                  0.4215
Relative absolute error                 72.5934 %
Root relative squared error             84.2924 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.850    0.390    0.685      0.850    0.759      0.474    0.824     0.799     positive
                 0.610    0.150    0.803      0.610    0.693      0.474    0.824     0.798     negative
Weighted Avg.    0.730    0.270    0.744      0.730    0.726      0.474    0.824     0.799

=== Confusion Matrix ===

   a   b   <-- classified as
 850 150 |   a = positive
 390 610 |   b = negative
```

Time taken to build model: 0.01 seconds.

CLASSIFICATION ATTEMPT II: Sequential Minimization Optimization

weka.classifiers: functions.SMO, BinarySMO

SMO Synopsis:
Implements John Platt's sequential minimal optimization algorithm for training a support vector classifier. This implementation globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes by default. To obtain proper probability estimates, use the option that fits calibration models to the outputs of the support vector machine..

OPTIONS:

- **Default**

Attempt II - Results

```
=== Summary ===

Correctly Classified Instances        1703               85.15   %
Incorrectly Classified Instances       297               14.85   %
Kappa statistic                          0.703
Mean absolute error                      0.1485
Root mean squared error                  0.3854
Relative absolute error                 29.7    %
Root relative squared error             77.0714 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.848    0.145    0.854      0.848    0.851      0.703    0.852     0.800     positive
                 0.855    0.152    0.849      0.855    0.852      0.703    0.852     0.798     negative
Weighted Avg.    0.852    0.149    0.852      0.852    0.851      0.703    0.852     0.799

=== Confusion Matrix ===

   a   b   <-- classified as
 848 152 |   a = positive
 145 855 |   b = negative
```

Time taken to build model: 2.99 seconds.

CLASSIFICATION ATTEMPT III: Naive-Bayes

weka.classifiers: bayes.NaiveBayes **(**Naive Bayes Classifier)

Naive-Bayes Synopsis:
Class for a Naive Bayes classifier using estimator classes. Numeric estimator precision values are chosen based on analysis of the training data.

OPTIONS:
- **Default**

Attempt III - Results

```
=== Summary ===

Correctly Classified Instances        1639               81.95   %
Incorrectly Classified Instances       361               18.05   %
Kappa statistic                          0.639
Mean absolute error                      0.1811
Root mean squared error                  0.418
Relative absolute error                 36.2132 %
Root relative squared error             83.6055 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.814    0.175    0.823      0.814    0.819      0.639    0.870     0.846     positive
                 0.825    0.186    0.816      0.825    0.820      0.639    0.868     0.836     negative
Weighted Avg.    0.820    0.181    0.820      0.820    0.819      0.639    0.869     0.841

=== Confusion Matrix ===

   a   b   <-- classified as
 814 186 |   a = positive
 175 825 |   b = negative
```

Time taken to build model: 0.18 seconds.

CLASSIFICATION ATTEMPT IV: Multinomial Naive-Bayes

weka.classifiers: bayes.NaiveBayesMultinomial

Multinomial Naive-Bayes Synopsis:
Class for building and using a multinomial Naive Bayes classifier. The core equation for this classifier: $P[C_i|D] = (P[D|C_i] \times P[C_i]) / P[D]$ (Bayes' rule) . where C_i is class i and D is a document.

OPTIONS:
• **Default**

Attempt IV - Results

```
=== Summary ===

Correctly Classified Instances        1700               85      %
Incorrectly Classified Instances       300               15      %
Kappa statistic                          0.7
Mean absolute error                      0.1595
Root mean squared error                  0.3558
Relative absolute error                 31.9034 %
Root relative squared error             71.1629 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.845    0.145    0.854      0.845    0.849      0.700    0.926     0.925     positive
                 0.855    0.155    0.847      0.855    0.851      0.700    0.926     0.926     negative
Weighted Avg.    0.850    0.150    0.850      0.850    0.850      0.700    0.926     0.926

=== Confusion Matrix ===

   a   b   <-- classified as
 845 155 |   a = positive
 145 855 |   b = negative
```

Time taken to build model: 0.07 seconds.

Additional Notes:
The independent probability of a class
----
positive 0.5
negative 0.5
The probability of a word given the class

CLASSIFICATION ATTEMPT V: Multinomial Logistic Regression

weka.classifiers: functions.Logistic

Multinomial Logistic Regression Synopsis:
Class for building and using a multinomial logistic regression model with a ridge estimator.

OPTIONS:

- **Default**

Attempt V - Results

```
=== Summary ===

Correctly Classified Instances       1713               85.65   %
Incorrectly Classified Instances      287               14.35   %
Kappa statistic                         0.713
Mean absolute error                     0.1737
Root mean squared error                 0.3274
Relative absolute error                34.7488 %
Root relative squared error            65.484  %
Total Number of Instances            2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.859    0.146    0.855      0.859    0.857      0.713    0.930     0.925     positive
                 0.854    0.141    0.858      0.854    0.856      0.713    0.930     0.929     negative
Weighted Avg.    0.857    0.144    0.857      0.857    0.856      0.713    0.930     0.927

=== Confusion Matrix ===

   a   b   <-- classified as
 859 141 |   a = positive
 146 854 |   b = negative
```

Time taken to build model: 0.8 seconds.

CLASSIFICATION ATTEMPT VI: Multilayer Perceptron

weka.classifiers: functions.MultilayerPerceptron

Multilayer Perceptron Synopsis:
A classifier that uses backpropagation to learn a multi-layer perceptron to classify instances. The network can be built by hand or or set up using a simple heuristic. The network parameters can also be monitored and modified during training time. The nodes in this network are all sigmoid.

OPTIONS:
- **Default**

Attempt VI - Results

```
=== Summary ===

Correctly Classified Instances        1629               81.45   %
Incorrectly Classified Instances       371               18.55   %
Kappa statistic                          0.629
Mean absolute error                      0.1883
Root mean squared error                  0.4077
Relative absolute error                 37.6649 %
Root relative squared error             81.5447 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.848    0.219    0.795      0.848    0.821      0.630    0.900     0.900     positive
                 0.781    0.152    0.837      0.781    0.808      0.630    0.900     0.905     negative
Weighted Avg.    0.815    0.186    0.816      0.815    0.814      0.630    0.900     0.903

=== Confusion Matrix ===

   a   b   <-- classified as
 848 152 |   a = positive
 219 781 |   b = negative
```

Time taken to build model: 366.09 seconds.

Additional Notes:
Ten-fold Cross validation took 82 minutes to complete.

CLASSIFICATION ATTEMPT VII: Stochastic Gradient Descent

weka.classifiers: functions.SGD, (with Loss function: Hinge loss (SVM))

SGD Synopsis:
Implements stochastic gradient descent for learning various linear models (binary class SVM, binary class logistic regression, squared loss, Huber loss and epsilon-insensitive loss linear regression). Globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes, so the coefficients in the output are based on the normalized data. For numeric class attributes, the squared, Huber or epsilon-insensitive loss function must be used. Epsilon-insensitive and Huber loss may require a much higher learning rate.

OPTIONS:

- **Default**

Attempt VII - Results

```
=== Summary ===

Correctly Classified Instances        1713               85.65   %
Incorrectly Classified Instances       287               14.35   %
Kappa statistic                          0.713
Mean absolute error                      0.1435
Root mean squared error                  0.3788
Relative absolute error                 28.7    %
Root relative squared error             75.7628 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.863    0.150    0.852      0.863    0.857      0.713    0.857     0.804     positive
                 0.850    0.137    0.861      0.850    0.856      0.713    0.857     0.807     negative
Weighted Avg.    0.857    0.144    0.857      0.857    0.856      0.713    0.857     0.805

=== Confusion Matrix ===

   a   b   <-- classified as
 863 137 |   a = positive
 150 850 |   b = negative
```

Time taken to build model: 0.89 seconds.

CLASSIFICATION ATTEMPT VIII: Linear Logistic Regression

weka.classifiers: functions.SimpleLogistic, (SimpleLogistic)

Simple Logistic Synopsis:
Classifier for building linear logistic regression models. LogitBoost with simple regression functions as base learners is used for fitting the logistic models. The optimal number of LogitBoost iterations to perform is cross-validated, which leads to automatic attribute selection.

OPTIONS:

- **Default**

Attempt VIII - Results

```
=== Summary ===

Correctly Classified Instances        1712               85.6    %
Incorrectly Classified Instances       288               14.4    %
Kappa statistic                          0.712
Mean absolute error                      0.1901
Root mean squared error                  0.3197
Relative absolute error                 38.0192 %
Root relative squared error             63.9454 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.857    0.145    0.855      0.857    0.856      0.712    0.933     0.934     positive
                 0.855    0.143    0.857      0.855    0.856      0.712    0.933     0.932     negative
Weighted Avg.    0.856    0.144    0.856      0.856    0.856      0.712    0.933     0.933

=== Confusion Matrix ===

   a   b   <-- classified as
 857 143 |   a = positive
 145 855 |   b = negative
```

Time taken to build model: 5.82 seconds.

CLASSIFICATION ATTEMPT VIIII: Voted Perceptron

weka.classifiers: functions.VotedPerceptron, (Number of perceptrons=427)

Voted Perceptron Synopsis:
Implementation of the voted perceptron algorithm by Freund and Schapire. Globally replaces all missing values, and transforms nominal attributes into binary ones.

OPTIONS:

- **Default**

Attempt VIIII - Results

```
=== Summary ===

Correctly Classified Instances        1688               84.4    %
Incorrectly Classified Instances       312               15.6    %
Kappa statistic                          0.688
Mean absolute error                      0.156
Root mean squared error                  0.395
Relative absolute error                 31.2018 %
Root relative squared error             78.9937 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.847    0.159    0.842      0.847    0.844      0.688    0.845     0.792     positive
                 0.841    0.153    0.846      0.841    0.844      0.688    0.871     0.818     negative
Weighted Avg.    0.844    0.156    0.844      0.844    0.844      0.688    0.858     0.805

=== Confusion Matrix ===

   a   b   <-- classified as
 847 153 |   a = positive
 159 841 |   b = negative
```

Time taken to build model: 0.23 seconds.

CLASSIFICATION ATTEMPT X: Random Forest

weka.classifiers: tree.RandomForest, (Bagging with 100 iterations and base learner)

Random Forest Synopsis:
Class for constructing a forest of random trees.

OPTIONS:
- **Default**

Attempt X - Results

```
=== Summary ===

Correctly Classified Instances        1643               82.15   %
Incorrectly Classified Instances       357               17.85   %
Kappa statistic                          0.643
Mean absolute error                      0.3689
Root mean squared error                  0.3944
Relative absolute error                 73.778  %
Root relative squared error             78.8724 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.828    0.185    0.817      0.828    0.823      0.643    0.901     0.895     positive
                 0.815    0.172    0.826      0.815    0.820      0.643    0.901     0.900     negative
Weighted Avg.    0.822    0.179    0.822      0.822    0.821      0.643    0.901     0.897

=== Confusion Matrix ===

   a   b   <-- classified as
 828 172 |   a = positive
 185 815 |   b = negative
```

Time taken to build model: 4.51 seconds.

CLASSIFICATION ATTEMPT XI: Stochastic Gradient Descent

weka.classifiers: functions.SGD, (Loss function: Log loss (logistic regression))

SGD Synopsis:
Implements stochastic gradient descent for learning various linear models (binary class SVM, binary class logistic regression, squared loss, Huber loss and epsilon-insensitive loss linear regression). Globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes, so the coefficients in the output are based on the normalized data. For numeric class attributes, the squared, Huber or epsilon-insensitive loss function must be used. Epsilon-insensitive and Huber loss may require a much higher learning rate.

OPTIONS:
- **Loss function: Log loss (logistic regression)**

Attempt XI - Results

```
=== Summary ===

Correctly Classified Instances        1717               85.85   %
Incorrectly Classified Instances       283               14.15   %
Kappa statistic                          0.717
Mean absolute error                      0.1772
Root mean squared error                  0.325
Relative absolute error                 35.4475 %
Root relative squared error             65.0072 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.860    0.143    0.857      0.860    0.859      0.717    0.932     0.932     positive
                 0.857    0.140    0.860      0.857    0.858      0.717    0.932     0.932     negative
Weighted Avg.    0.859    0.142    0.859      0.859    0.858      0.717    0.932     0.932

=== Confusion Matrix ===

   a   b   <-- classified as
 860 140 |   a = positive
 143 857 |   b = negative
```

Time taken to build model: 0.91 seconds.

CLASSIFICATION ATTEMPT XII: Stochastic Gradient Descent

weka.classifiers: functions.SGD, (Loss function: Log loss (logistic regression))

SGD Synopsis:
Implements stochastic gradient descent for learning various linear models (binary class SVM, binary class logistic regression, squared loss, Huber loss and epsilon-insensitive loss linear regression). Globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes, so the coefficients in the output are based on the normalized data. For numeric class attributes, the squared, Huber or epsilon-insensitive loss function must be used. Epsilon-insensitive and Huber loss may require a much higher learning rate.

OPTIONS:

- **Loss function: *Log loss (logistic regression)***
- **Epoch: *800***
- **learning rate: *0.001***

Attempt XII - Results

```
=== Summary ===

Correctly Classified Instances        1722               86.1    %
Incorrectly Classified Instances       278               13.9    %
Kappa statistic                          0.722
Mean absolute error                      0.1919
Root mean squared error                  0.315
Relative absolute error                 38.388  %
Root relative squared error             62.9922 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.863    0.141    0.860      0.863    0.861      0.722    0.937     0.936     positive
                 0.859    0.137    0.862      0.859    0.861      0.722    0.937     0.939     negative
Weighted Avg.    0.861    0.139    0.861      0.861    0.861      0.722    0.937     0.938

=== Confusion Matrix ===

   a   b   <-- classified as
 863 137 |   a = positive
 141 859 |   b = negative
```

Time taken to build model: 1.35 seconds.

CLASSIFICATION ATTEMPT XIII: Stochastic Gradient Descent

weka.classifiers: functions.SGD, (Loss function: Log loss (logistic regression))

SGD Synopsis:
Implements stochastic gradient descent for learning various linear models (binary class SVM, binary class logistic regression, squared loss, Huber loss and epsilon-insensitive loss linear regression). Globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes, so the coefficients in the output are based on the normalized data. For numeric class attributes, the squared, Huber or epsilon-insensitive loss function must be used. Epsilon-insensitive and Huber loss may require a much higher learning rate.

OPTIONS:
- **Loss function: *Log loss (logistic regression)***
- **Epoch: *800***
- **learning rate: *0.0005***

Attempt XIII - Results

```
=== Summary ===

Correctly Classified Instances        1724               86.2    %
Incorrectly Classified Instances       276               13.8    %
Kappa statistic                          0.724
Mean absolute error                      0.2045
Root mean squared error                  0.3142
Relative absolute error                 40.9094 %
Root relative squared error             62.8371 %
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.864    0.140    0.861      0.864    0.862      0.724    0.938     0.937     positive
                 0.860    0.136    0.863      0.860    0.862      0.724    0.938     0.941     negative
Weighted Avg.    0.862    0.138    0.862      0.862    0.862      0.724    0.938     0.939

=== Confusion Matrix ===

   a   b   <-- classified as
 864 136 |   a = positive
 140 860 |   b = negative
```

Time taken to build model: 1.59 seconds.

v) [**20** points] Report the best method with its parameter(s) you have found including the performance-evaluation matrices. Explain, why do you think your selected top method is the best method out of the 10 methods you tried.

STEP 5: RESULTS - MODEL COMPARISONS

Rank	Classifier	Attempt #	Accuracy
1	**Stochastic Gradient Descent**	**XIII (13)**	**86.20%**
2	Stochastic Gradient Descent	XII (12)	86.10%
3	Stochastic Gradient Descent	XI (11)	85.85%
4	Stochastic Gradient Descent	VII (7)	85.65%
5	Multinomial Logistic Regression	V (5)	85.65%
6	Linear Logistic Regression	VIII (8)	85.60%
7	Sequential Minimization Optimization	II (2)	85.15%
8	Multinomial Naive-Bayes	IV (4)	85.00%
9	Voted Perceptron	VIIII (9)	84.40%
10	Random Forest	X (10)	82.15%
11	Naive-Bayes	III (3)	81.95%
12	Multilayer Perceptron	VI (6)	81.45%
13	k-Nearest Neighbors	I (1)	73.00%

For complete details regarding the performance of each model listed above, See: previous section.

Top Ranked Model: Stochastic Gradient Descent
The best performing predictor was a customized Stochastic Gradient Descent (SGD). the fine tuned versions outperformed the default mode. However, the default SGD out performed all of the other models, with the exception of Multinomial Logistic Regression (MLR), which scored similar. But MLR did not improve with additional fine tuning while SGD did. The top seven performing classifiers share one commonality, they all utilised a logistic function as the basis for their evaluations.

The conclusion from this test implies that Logistic functions work best for binary categorical data predictions. The behavior of a log function helps understand why this may be the case. Determining which class to select from a binary pair, makes sense to have a function that quickly defines a decision boundary line between the two classes, in this case, the negative class and the positive class. The marginal improvements made by fine tuning were just the result of adjusting the learning speed and the epoch count.

vi) [**20** points] Review literature to explain '**infoGainAttributeEval**' in details and cite the relevant reference(s). Submit the copies of the paper(s) that you have cited to explain '**infoGainAttributeEval**'.

According to the official Weka API documentation[1] *infoGainAttributeEva*l is:

```
InfoGainAttributeEval :

Evaluates the worth of an attribute by measuring the information gain with respect to
the class.

InfoGain(Class,Attribute) = H(Class) - H(Class | Attribute).
Valid options are:

 -M
  treat missing values as a separate value.
 -B
  just binarize numeric attributes instead
  of properly discretizing them.
```

Within the official Weka documentation, Mark Hall, author of infoGainAttributeEval cites that this implementation is based on the research from:

Usama M. Fayyad, Keki B. Irani: Multi-interval discretization of continuous valued attributes for classification learning. In: Thirteenth International Joint Conference on Artificial Intelligence, 1022-1027, 1993.

Igor Kononenko: On Biases in Estimating Multi-Valued Attributes. In: 14th International Joint Conference on Artificial Intelligence, 1034-1040, 1995

[1] InfoGainAttributeEval, Weka API Documentation Revision: 10172 ,Mark Hall
http://weka.sourceforge.net/doc.stable-3-8/weka/attributeSelection/InfoGainAttributeEval.html

A more detailed explanation for Information Gain and an example case study is given below.

InfoGainAttributeEval is used for **feature selection** tasks. What *InfoGainAttributeEval* basically does is measuring how each feature contributes in *decreasing the overall entropy*. Let's take an example. Say we have this dataset :

Temperature	Wind	Class
high	low	play
low	low	play
high	low	play
low	high	cancelled
low	low	play
high	high	canceled
high	low	play

The Entropy is defined as follows :

1 Entropy

Let $\mathbf{X}$ be a random variable: $P(\mathbf{X}=x)=p(x)$. Note that $\sum_{x \in X} p(x)=1$. The binary *Entropy* of random variable $\mathbf{X}$ is defined as:

$$H_2(\mathbf{X}) = -\sum_{x \in X} p(x)\ log_2\ p(x) \qquad bits. \tag{1}$$

As an example consider a *coin toss*. Let $P(H)=p$, and $P(T)=1-p$, such that $P(H)+P(T)=1$. The entropy of the coin toss:

$$H_2(p) = -p\ log_2\ p\ \ -(1-p)\ log_2\ (1-p). \tag{2}$$

When $p=0$: $H_2(p) = -0\ log_2\ 0\ \ -1\ log_2\ 1\ \ = 0.$
When $p=1$: $H_2(p) = -1\ log_2\ 1\ \ -0\ log_2\ 0\ \ = 0.$

$$H(x)\ =\ -\Sigma(P_i\, log_2(P_i))$$

,with P_i being the probability of the class i in the dataset, and log_2 the base 2 logarithm (in Weka natural logarithm of base *e* is used, but generally we take log2). Entropy basically measures the **degree of "impurity"**. The closest to 0 it is, the less impurity there is in your dataset. Hence, a good attribute is an attribute that **contains the most information**, i.e, **reduces the most the entropy**. The InfoGainAttributeEval method of Weka is a way of evaluating exactly this.

Now, the entropy of our example is :

H(Class) = -(5/7*log2(5/7)+2/7*log(2/7)) = 0,863.

Let's calculate for our example the amount of information carried by the temperature attribute.

InfoGain(Class,Temperature) = H(Class) - H(Class | Temperature).

To get the H(Class | Temperature), we need to split the dataset according to this attribute.

```
                     Dataset
                      /   \
                     /     \
                    / Temp° \
                   /         \
                  /           \
                 /             \
               high            low



    temperature | wind | class                 temperature|wind|class
high         | low  | play                     low     | low|play
high         | low  | play                     low     | high|cancelled
high         | high | cancelled                low     | low |play
high         | low  | play
```

Each branch here has its own entropy. We need to first calculate the entropy of each split.

$$H(leftside) = -\left(\frac{3}{4}\, log_2\left(\frac{3}{4}\right) + \frac{1}{4}\, log_2\left(\frac{1}{4}\right) \right) = 0.811$$

$$H(rightside) = -\left(\frac{1}{3}\, log_2\left(\frac{1}{3}\right) + \frac{2}{3}\, log_2\left(\frac{2}{3}\right) \right) = 0.918$$

H(Class | Temperature) is then equals to the sum of both children's entropy, weighted by the proportion of instances that where taken from the parent dataset. In short :

$$H(Class \mid Temperature) = \frac{4}{7} H(leftside) + \frac{3}{7} H(rightside)$$

You then have everything to calculate the InfoGain. In this example, it's 0,06 bits. This means that the temperature feature only reduces the global entropy by 0,06 bits, the **feature's contribution to reduce the entropy** (= the **information gain**) is fairly small.

This is pretty obvious looking at the instances in the dataset, as we can see at a first glance that the temperature doesn't affect much the final class, unlike the wind feature.

Sources :
KevinD, Article: How the selection happens in 'InfoGainAttributeEval' in weka feature selection (filter method), Stack Overflow, 2016; https://stackoverflow.com/questions/33982943/how-the-selection-happens-in-infogainattributeeval-in-weka-feature-selection

Anuj Sharma and Shubhamoy Dey. Article: Performance Investigation of Feature Selection Methods and Sentiment Lexicons for Sentiment Analysis. IJCA Special Issue on Advanced Computing and Communication Technologies for HPC Applications ACCTHPCA(3):15-20, July 2012

Appendix 1: *Weka Preprocess Data - Time Capture*

System-level Python Script

```
"""
Requires: Weka JAR file, JRE, Python 2.7
Instructions:
    1. Set the following environmentals: wekaJAR, src, dest
    2. run the script
"""
import time
import os

wekaJAR = '' #BASH command to find your Weka JAR path: find / -name \weka.jar
src = ''     #Source directory of intial Dataset
dest = './'  #Destination directory for output: ARFF file

#Captures Runtime of WEKA dataset conversion into ARFF
def main():
    msBefore = time.time()*1000.0
    getARFF(wekaJAR, src, dest);
    msAfter = time.time()*1000.0
    print str(msAfter - msBefore) + " milliseconds"

#WEKA CLI: convert dataset into ARFF file format
def getARFF(wekaJAR, src, dest):
    className = 'weka.core.converters.TextDirectoryLoader';
    dest += 'ProcessedData.arff';
    bashCMD = 'java -cp {weka} {className} -dir {src} > {dest}';
    bashCMD = bashCMD.format(weka=wekaJAR, className=className, src=src, dest=dest);
    os.system(bashCMD);


if __name__ == '__main__': main()
```

Appendix 2: Sources

Usama M. Fayyad, Keki B. Irani: Multi-interval discretization of continuous valued attributes for classification learning. In: Thirteenth International Joint Conference on Artificial Intelligence, 1022-1027, 1993.

Anuj Sharma and Shubhamoy Dey. Article: Performance Investigation of Feature Selection Methods and Sentiment Lexicons for Sentiment Analysis. IJCA Special Issue on Advanced Computing and Communication Technologies for HPC Applications ACCTHPCA(3):15-20, July 2012

KevinD, Article: How the selection happens in 'InfoGainAttributeEval' in weka feature selection (filter method), Stack Overflow, 2016; https://stackoverflow.com/questions/33982943/how-the-selection-happens-in-infogainattributeeval-in-weka-feature-selection