Assignment 2 - Using Weka for Text Classification

Matthew Acs

February 18, 2024

1 Introduction

The objective of this assignment is to explore text classification using Weka, a popular machine learning toolkit. The aim is to classify documents into predefined categories using a preprocessed document collection. Specifically, the Naïve Bayes algorithm implemented in Weka will be utilized to conduct text classification. The dataset provided for this task is the WebKB dataset, consisting of web pages collected from computer science departments of various universities. These web pages are classified into four categories: student, faculty, project, and course. The dataset has been preprocessed by removing stop words and stemming, and it is provided in the .arff format for direct import into Weka. The primary goal is to evaluate the classification accuracy of the Naïve Bayes algorithm on this dataset and provide insights into its performance for text classification tasks.

2 Data

The first step is to get an understanding of the data. Upon opening the training data file (Figure 1) it can be seen that the data is already preprocessed. All words are lowercase and stemmed. Additionally, each web page text body is contained within single quotes, which is followed by a comma and the class it belongs to. There are four possible classes (student, faculty, project, and course). The training and testing data files are arff files, which are specific to Weka. Finally, after loading the data into Weka, it can be seen that the classification problem is imbalanced (Table 1). The student class has the most data, followed by the faculty, course, and finally project. This trend is shown in both the training and testing data in similar ratios. The largest class contains 39% of the testing data, thus a naïve classifier that always chooses the largest class will have an accuracy of 39%. Additionally, a random classifier will achieve a 25% accuracy.

Table 1: Data Count

Training Web Page Count by Class	
Student	1097
Faculty	750
Project	336
Course	620
Testing Web Page Count by Class	
Student	544
Faculty	374
Project	168
Course	310

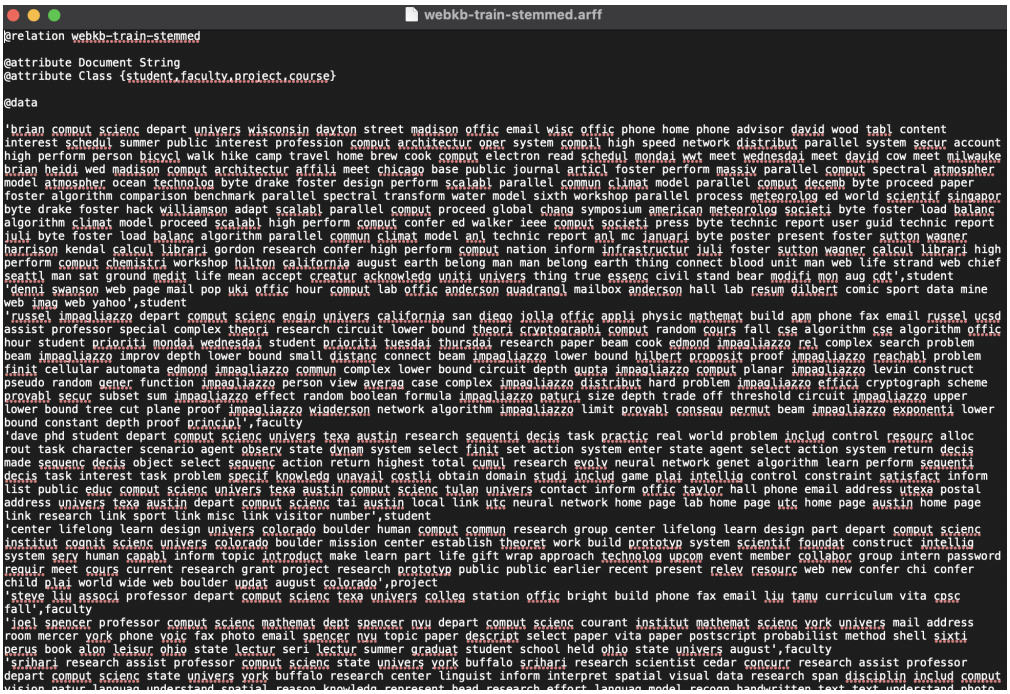


Figure 1: Training data file

3 Methodology

Now that we have a basic understanding of the data, we can train a classifier to predict the type of webpage. This section will describe the steps taken to prepare the data for classification as well as the steps taken to train a Naïve Bayes classifier.

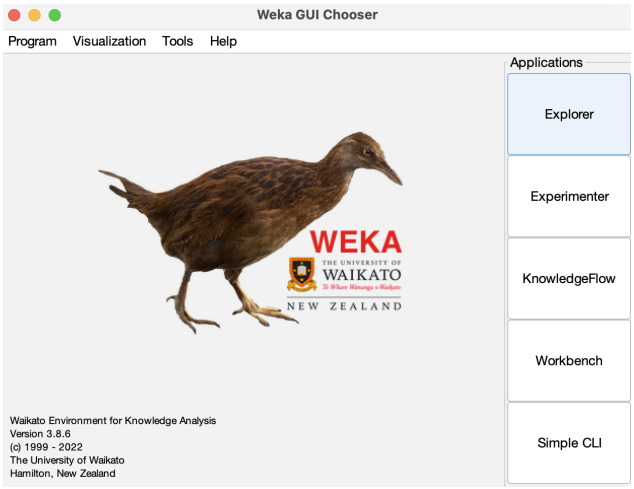


Figure 2: Weka after installation

3.1 Data Loading

The first step involves loading the preprocessed dataset into Weka. Once Weka is installed (Figure 2), the Explorer application can be accessed. Here, the data file can be opened (Figure 3), which will show the data

in the explorer along with a class count visualization. Only the training data was initially loaded in the preprocessing tab, as the testing data and training data will later go through a preprocessing-classification pipeline, thus any work done in the next step will be redone during classification. Figure 3 shows the class imbalance that was discussed in the data section of this report.

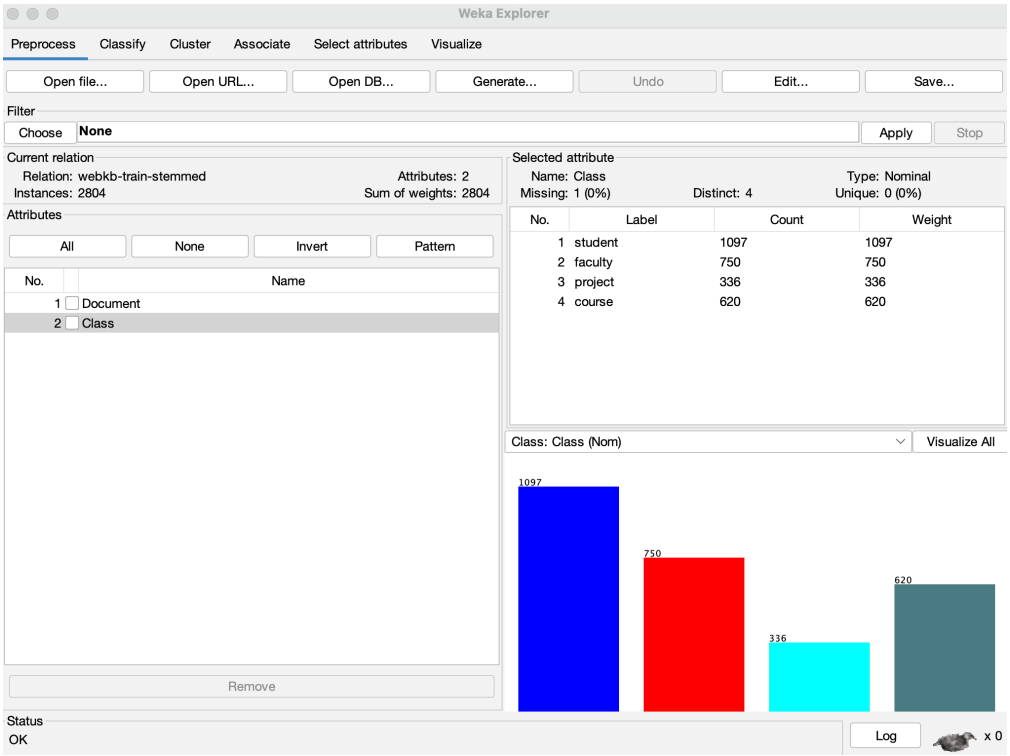


Figure 3: Weka explorer after loading training data

3.2 Generating Document-Word Matrices

Once the dataset was loaded, the document-word matrices were generated to represent the text data in a structured format suitable for machine learning analysis. These matrices encode the frequency of occurrence of each word in each document, providing a numerical representation of the textual data. To do this in Weka, the StringToWordVector filter was chosen with the settings shown in Figure 4. The vectorizer creates the document-word matrices along with TF-IDF transformations. Figure 5 shows the result of the text vectorization in the preprocess tab of the explorer. Clicking the edit button allows us to see the full TF-IDF document word matrix (Figure 6). Here the class attribute can also be specified, which results in the Weka explorer showing the class distribution of each word (Figure 7). Finally, Figure 8 shows the data file after vectorization. This data file now represents the document-word matrix and is ready for classification. Weka allows for the creation of a preprocess-classification pipeline that applies the same preprocessing operations to both the testing and training datasets. This can be done under the classification tab directly to the raw data, and thus all the steps completed in sectoin 3.2 will be reverted to the result from section 3.1 in order to proceed to section 3.3.

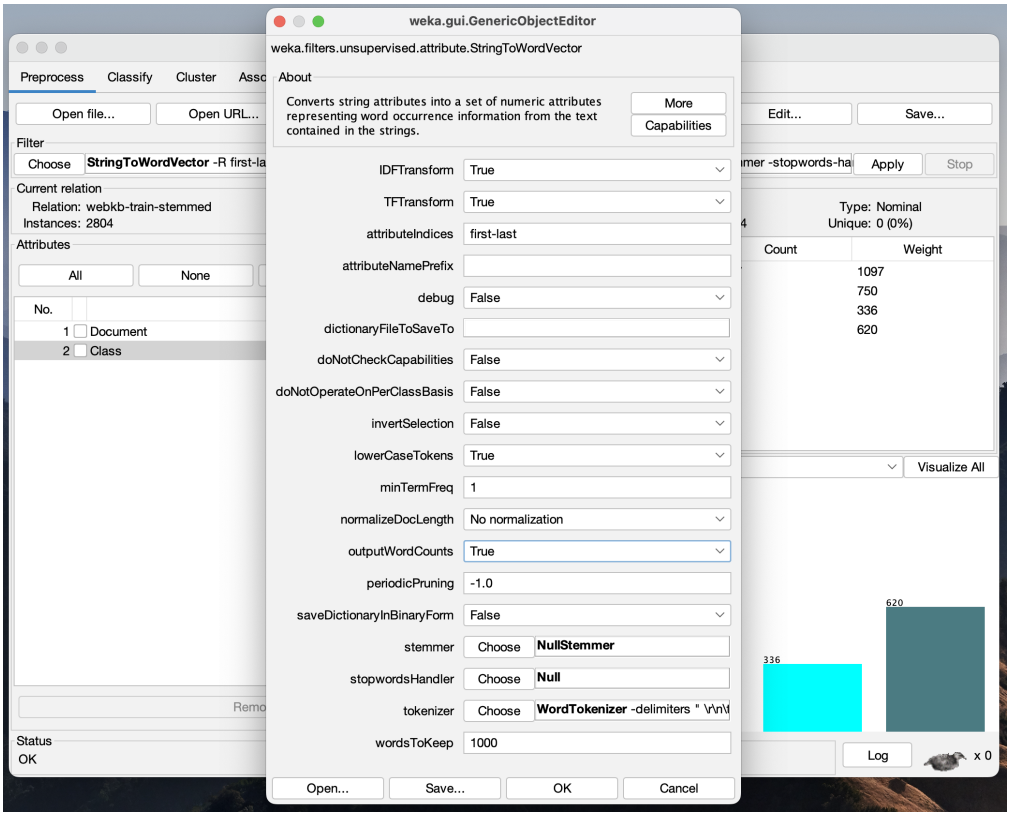


Figure 4: Setting up the text vectorizer/tokenizer

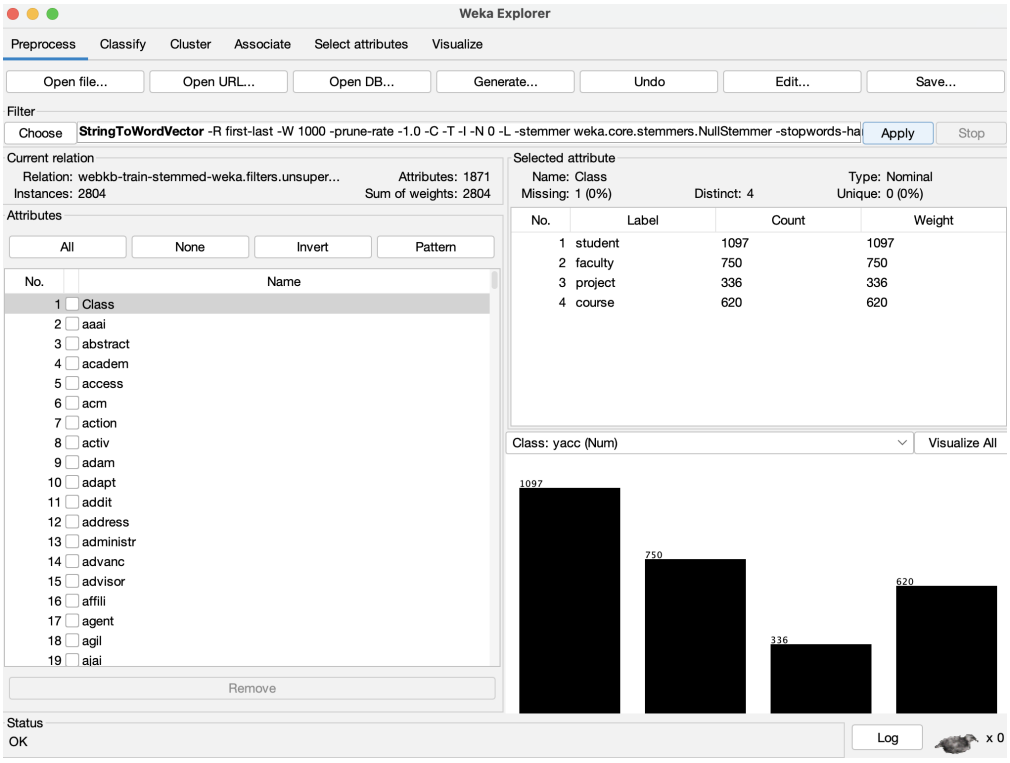


Figure 5: Data after vectorization

Viewer

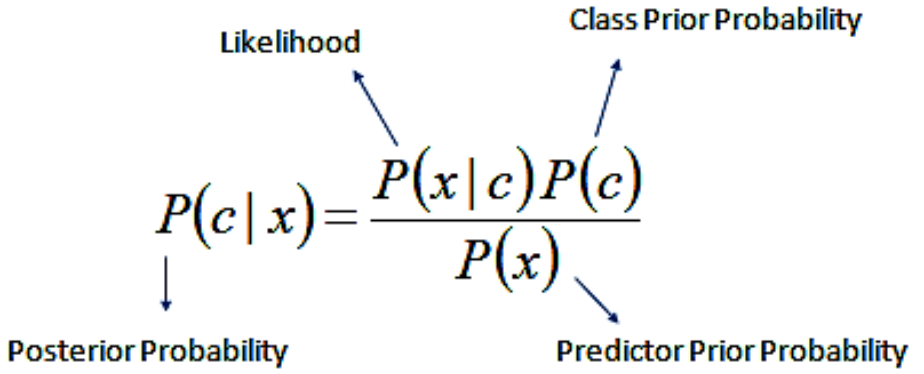
Relation: webkb-train-stemmed-weka.filters.unsupervised.attribute.StringToWordVector-R1-W1000-prune-rate-1.0-C-T-I-N0-L-stemmerweka.core.stemmers.NullStemmer..

No.	1: Class Nominal	2: aaai Numeric	3: abstract Numeric	4: academ Numeric	5: access Numeric	6: acm Numeric	7: action Numeric	8: activ Numeric	9: adam Numeric	10: adapt Numeric	11: addit Numeric	12: address Numeric	13: administr Numeric	14: advanc Numeric	15: a Nu
1	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.14477...	0.0	0.0	0.0	0.0	1.88
2	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	student	0.0	0.0	0.0	0.0	0.0	5.6973...	0.0	0.0	0.0	0.0	2.0047074...	0.0	0.0	
4	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
5	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
6	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
7	student	0.0	0.0	0.0	1.55358...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
8	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2648295...	0.0	0.0	
9	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.88
10	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2648295...	0.0	0.0	
11	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
12	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
13	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
14	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
15	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
16	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
17	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
18	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0047074...	0.0	0.0	
19	student	0.0	0.0	0.0	1.55358...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
20	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2648295...	0.0	0.0	
21	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
22	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2648295...	0.0	0.0	
23	student	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
24	student	0.0	0.0	0.0	1.55358...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

Add instance | Undo | OK | Cancel

Figure 6: Document-word matrix

Generating the document-word matrix was necessary to use the Naive Bayes classifier because the classifier needs a numeric representation of each data point. It is interesting to note that Weka includes another classifier called Naive Bayes Multinomial Text. This classifier operates directly on strings, and thus does not require the input to be vectorized.



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Figure 7: Naive Bayes

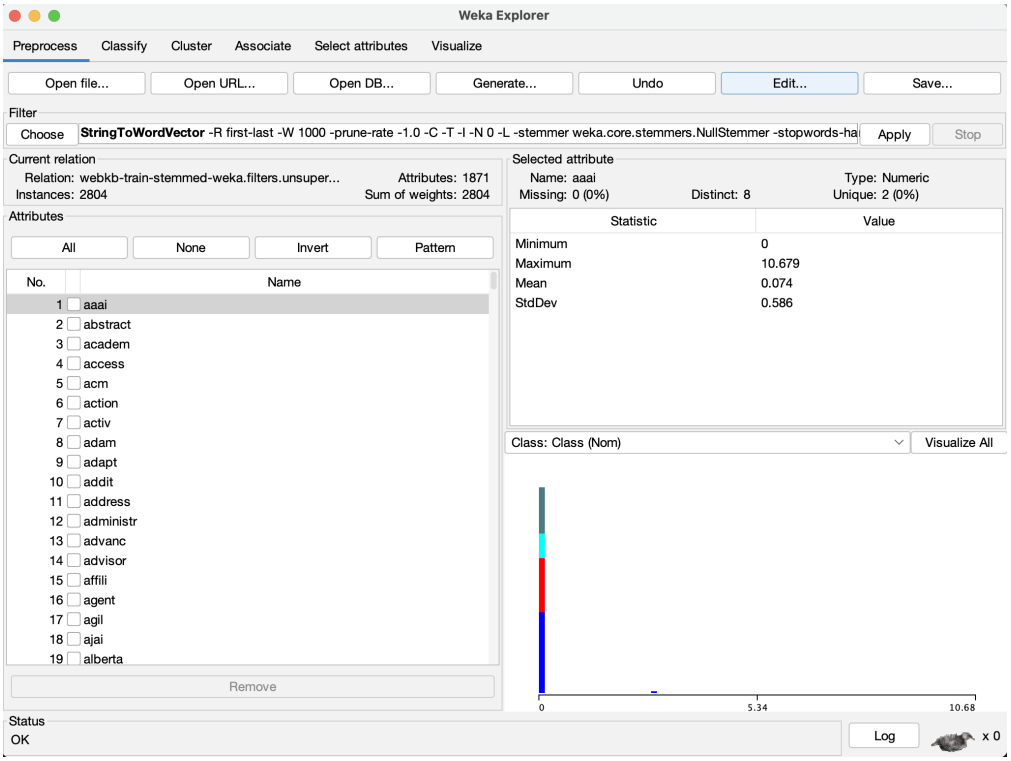


Figure 8: Data after specifying class attribute

Figure 9: Data file after vectorization

3.3 Classification Using Naïve Bayes

After preparing the data, the Naïve Bayes algorithm will be used to conduct text classification. Naïve Bayes is a probabilistic classifier that assigns class labels to instances based on the likelihood of features given each class. The Naïve Bayes classifier is trained on the training dataset and evaluated on the test dataset. The first step is to open the classify tab with the raw data loaded in the preprocess tab. Then the testing data is specified by choosing the supplied test set option, shown in Figure 10. Next, the classifier is chosen. The classifier was set to a filtered classifier with the filter being set to the StringToWordVector filter discussed in section 3.2 and the classifier being set to NaiveBayes (Figure 11). This classifier setup will preprocess the training and testing data according to the method discussed previously before training the Naive Bayes classifier. Pressing start begins the training process, which is output on the right. Figure 12 shows the classification results.

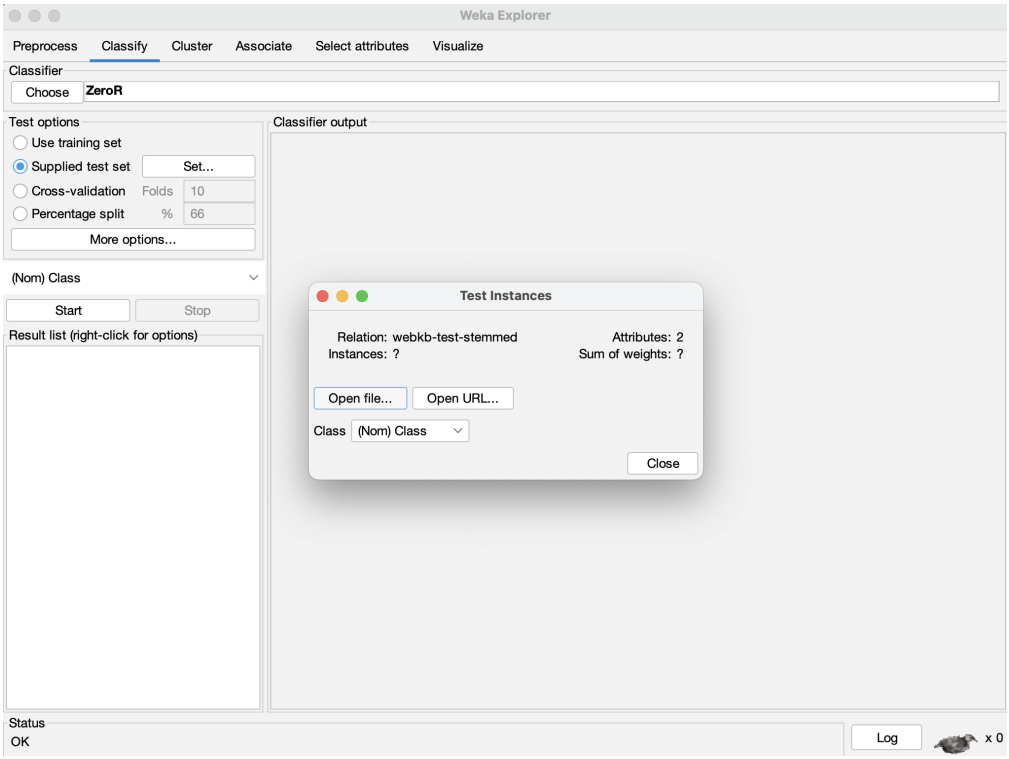


Figure 10: Loading the test data for classification

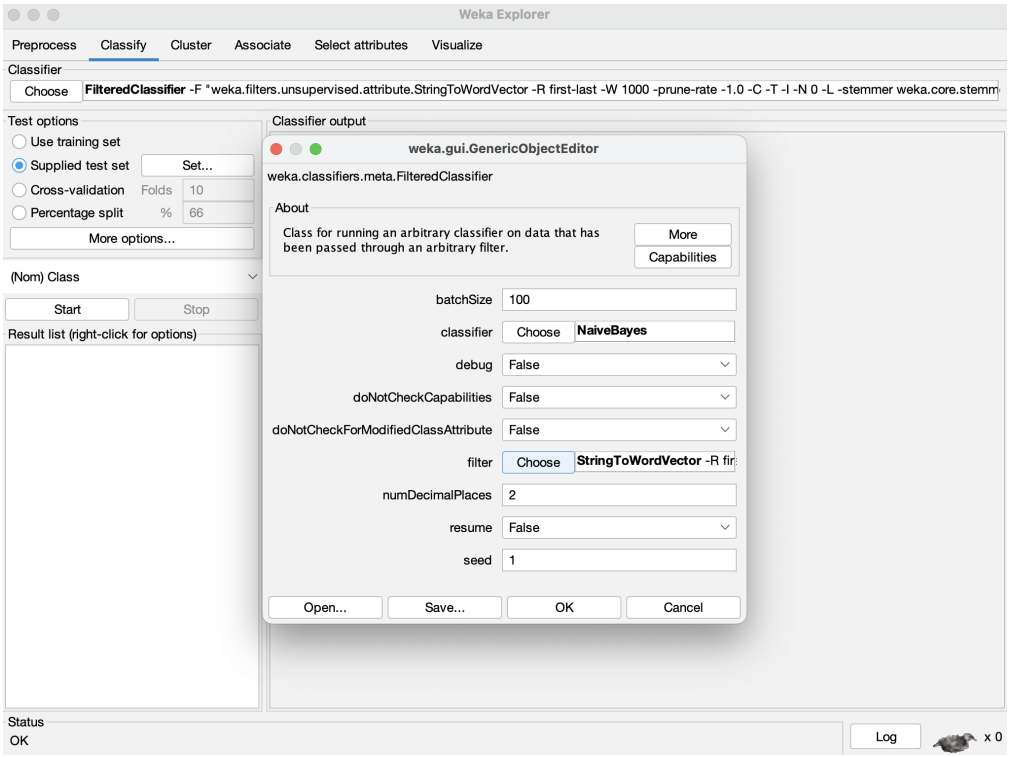


Figure 11: Setting up the classifier

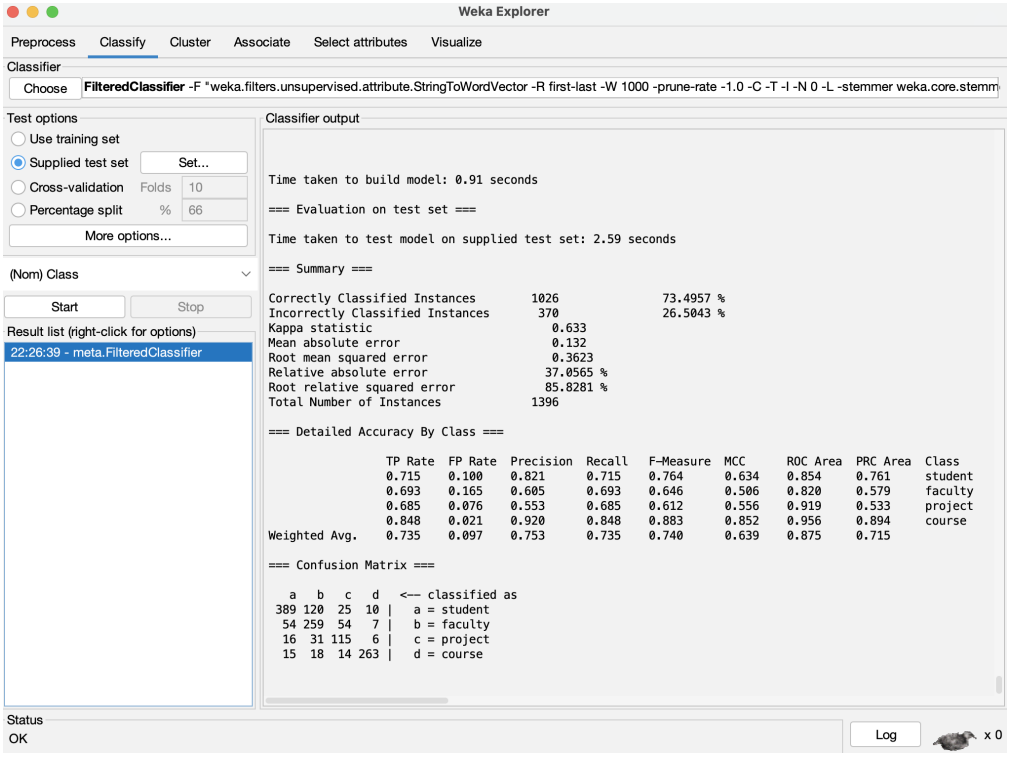


Figure 12: Classification results

4 Results

The classification results are shown in Table 2. The model was able to achieve a 73.5% classification accuracy, which is far greater than either the 39% or 25% accuracy posed by the theoretical naive and random approaches discussed earlier.

Table 2: Model Evaluation Results

Time taken to build model: 0.91 seconds

Evaluation on test set	
Time taken to test model on supplied test set	2.59 seconds

Summary	
Correctly Classified Instances	1026 (73.4957%)
Incorrectly Classified Instances	370 (26.5043%)
Kappa statistic	0.633
Mean absolute error	0.132
Root mean squared error	0.3623
Relative absolute error	37.0565%
Root relative squared error	85.8281%
Total Number of Instances	1396

Detailed Accuracy By Class									
Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	
student	0.715	0.100	0.821	0.715	0.764	0.634	0.854	0.761	
faculty	0.693	0.165	0.605	0.693	0.646	0.506	0.820	0.579	
project	0.685	0.076	0.553	0.685	0.612	0.556	0.919	0.533	
course	0.848	0.021	0.920	0.848	0.883	0.852	0.956	0.894	
Weighted Avg.	0.735	0.097	0.753	0.735	0.740	0.639	0.875	0.715	

Confusion Matrix				
	Classified as			
Actual	389	120	25	10
	54	259	54	7
	16	31	115	6
	15	18	14	263

5 Discussions and Conclusions

Overall, Weka was able to preprocess the data to create document-word matrix representations and train a Naive Bayes classifier. The classification accuracy was 73.5%, which is relatively successful. The ROC area and other classification results also show that the model was successful. Document-word matrix generation was also successful, as it transformed the data into TF-IDF vectors. The filter that was used to complete this operation had multiple options, allowing for a configuration that does not utilize TF-IDF weighting. Additionally, Weka's GUI made the process easy and intuitive. Rapid experimentation is simple with the drop down menu of models, algorithms, filters, and more. Weka also provides an easy ways to visualize different operations as you experiment with your data.

6 References

The following sources were referenced during the completion of this assignment.

1. https://www.youtube.com/watch?v=yQofi_4Z-lw
2. https://karim-ouda.medium.com/tutorial-document-classification-using-weka-aa98d5edb6fa
3. https://www.youtube.com/watch?v=IY29uC4uem8
4. https://afit-r.github.io/naive_bayes