IFN645: Large Scale Data Mining

Capstone Assignment

Student 1: Steve Ryenchindorj

Student ID: 11790300

Student 2: Nathan Edwards

Student ID: 11657464

Task 1: Association Mining in Java

A bank has conducted a marketing campaign via phone calls to promote their new products including a term-deposit product. After the campaign, the bank wants to analyse the data collected in the campaign in order to get a better understanding of their customers.

Question 1

Generate frequent patterns from the entire dataset using two frequent pattern mining algorithms and compare their performance in terms of time efficiency. Use different minimum supports (e.g. 0.2, 0.3 and 0.4) to do the comparison. Show your comparison result.

Algorithm	min_sup = 0.2	min_sup = 0.3	min_sup = 0.4
APRIORI	Total time ~ 1012 ms	Total time ~ 683 ms	Total time ~ 267 ms
FP-GROWTH	Total time ~ 357 ms	Total time ~ 270 ms	Total time ~ 108 ms

Question 2

Choose a frequent pattern mining algorithm based on the comparison in Question 1, state the reason to choose this algorithm.

Based on the comparisons in Question 1, FP-Growth is chosen as it is faster and more efficient in terms of both time and memory usage compared to Apriori, particularly with larger datasets. For example, with the minimum support set to 0.2, FP-Growth completed in 357ms, whereas Apriori took 1012ms to achieve the same number of frequent itemsets (401).

Use the chosen algorithm to generate the top 5 most frequent size-3 patterns from the yes-class dataset and the no-class dataset, separately, then compare the generated patterns from the two datasets. Do the customers in the two classes share any common characteristics? Are there any different characteristics between the customers in the two classes? List the common characteristics and differences, if there are any. Specify the minimum support that you have used for this question.

	Top 5 most frequent size-3 patterns for YES class (min_sup 0.4)	Top 5 most frequent size-3 patterns for NO class (min_sup 0.4)
1	default_credit=no housing=no loan=no #SUP: 3120	default_credit=no past_marketing=unknown loan=no #SUP: 27371
2	default_credit=no past_marketing=unknown loan=no #SUP: 2988	default_credit=no past_marketing=unknown call_duration=100-500s #SUP: 21352
3	default_credit=no loan=no call_duration=100-500s #SUP: 2712	default_credit=no loan=no call_duration=100-500s #SUP: 21217
4	default_credit=no loan=no age=21-30s #SUP: 2519	default_credit=no past_marketing=unknown marital=married #SUP: 20357
5	default_credit=no marital=married loan=no #SUP: 2471	default_credit=no marital=married loan=no #SUP: 19799

Upon inspection, there are two clear similarities between the yes-class and no-class. Individuals in these classes share the characteristic default_credit=no across the top 5 patterns and frequently do not have loans (loan=no). Characteristics of past_marketing=unknown, marital=married and call_duration=100-500s are shared, but more frequent in the no-class. The characteristic of age=21-30s is present only in the yes-class. Therefore, this group of customers are likely to engage positively. The key difference between the classes is the number of supporting patterns. The most frequent size-3 pattern in the yes-class appears 3120 times, while the top size-3 pattern in the no class appears 27,371 times in the dataset.

The minimum support is set at 0.4 for this question.

Question 3

Generate the top 5 most frequent maximum patterns from yes-class and no-class datasets separately, identify any similarity or difference between the two classes in terms of the maximum patterns. List the similar characteristics and differences, if there are any. Specify the minimum support that you have used for this question.

	Top 5 maximum patterns for YES class (min_sup 0.1)	**Top 5 maximum patterns for NO class (min_sup 0.1)**
1	default_credit=no housing=no education=tertiary loan=no call_duration=100-500s #SUP: 865	default_credit=no past_marketing=unknown loan=no job=blue-collar #SUP: 6142
2	default_credit=no housing=no loan=no education=secondary call_duration=100-500s #SUP: 851	default_credit=no past_marketing=unknown balance=below-1k housing=no loan=no #SUP: 6124
3	default_credit=no past_marketing=unknown balance=below-1k loan=no age=21-30s #SUP: 844	default_credit=no past_marketing=unknown marital=single loan=no age=21-30s #SUP: 6098
4	default_credit=no past_marketing=unknown housing=no education=tertiary loan=no #SUP: 799	default_credit=no past_marketing=unknown marital=married loan=no housing=yes call_duration=100-500s #SUP: 6062
5	default_credit=no housing=no education=tertiary loan=no age=21-30s #SUP: 783	default_credit=no past_marketing=unknown loan=no housing=yes call_duration=100-500s age=21-30s #SUP: 5783

The similar characteristics in both classes appear to include characteristics such as default_credit=no, loan=no, past_marketing=unknown, balance=below-1k, call_duration=100-500s, and age=21-30s. Inspection of the dataset, confirms that these attributes appear most frequently, and are expected to feature in the maximum patterns output. In terms of differences, the yes-class tend to have higher education and may not own housing, as opposed to the no-class who are more likely to be homeowners, in blue-collar jobs, and may have a marital/housing ownership link. These characteristic differences affirm the type of customers the campaign would be targeting. Those with a house and/or married and those with lower income jobs are less likely to have the need for new banking products. In contrast, those without homes and with higher education completion are more likely to be looking to get a loan for a house or invest money in a term-deposit.

The minimum support is set at 0.1 for this question. After trialling different minimum supports, 0.1 provided the most meaningful results that showed differences between the two classes.

Question 4

Using the entire dataset, generate the top 10 most frequent association rules with subscribed=yes as the consequent and also the top 10 most frequent association rules with subscribed=no as the consequent. Specify the minimum confidence used. Observe the rules and identify any redundant rules in each set of the rules. If there exist redundant rules, list them and state why you think they are redundant.

	Top 10 most frequent association rules with subscribed=YES as the consequent (minconf=0.8)	**Top 10 most frequent association rules with subscribed=NO as the consequent (minconf=0.8)**

1	past_marketing=success call_duration=500-1k balance=below-1k ==> subscribed=yes #SUP: 78 #CONF: 0.848	marital=married ==> subscribed=no #SUP: 24459 #CONF: 0.899
2	default_credit=no past_marketing=success call_duration=500-1k balance=below-1k ==> subscribed=yes #SUP: 78 #CONF: 0.848	default_credit=no call_duration=100-500s ==> subscribed=no #SUP: 25538 #CONF: 0.899
3	housing=no loan=no past_marketing=success call_duration=500-1k ==> subscribed=yes #SUP: 106 #CONF: 0.891	call_duration=100-500s ==> subscribed=no #SUP: 26044 #CONF: 0.900
4	default_credit=no housing=no loan=no past_marketing=success call_duration=500-1k ==> subscribed=yes #SUP: 106 #CONF: 0.891	default_credit=no loan=no past_marketing=unknown ==> subscribed=no #SUP: 27371 #CONF: 0.902
5	housing=no past_marketing=success call_duration=500-1k ==> subscribed=yes #SUP: 109 #CONF: 0.886	loan=no past_marketing=unknown ==> subscribed=no #SUP: 27814 #CONF: 0.902
6	default_credit=no housing=no past_marketing=success call_duration=500-1k ==> subscribed=yes #SUP: 109 #CONF: 0.886	default_credit=no loan=no ==> subscribed=no #SUP: 32685 #CONF: 0.873
7	loan=no past_marketing=success call_duration=500-1k ==> subscribed=yes #SUP: 154 #CONF: 0.824	default_credit=no past_marketing=unknown ==> subscribed=no #SUP: 32862 #CONF: 0.908
8	default_credit=no loan=no past_marketing=success call_duration=500-1k ==> subscribed=yes #SUP: 154 #CONF: 0.824	loan=no ==> subscribed=no #SUP: 33162 #CONF: 0.873
9	past_marketing=success call_duration=500-1k ==> subscribed=yes #SUP: 161 #CONF: 0.809	past_marketing=unknown ==> subscribed=no #SUP: 33573 #CONF: 0.908
10	default_credit=no past_marketing=success call_duration=500-1k ==> subscribed=yes #SUP: 161 #CONF: 0.813	default_credit=no ==> subscribed=no #SUP: 39159 #CONF: 0.882

When each association rule has the same consequent, a rule becomes redundant if there is a super rule or more comprehensive rule, that carries the more information in the antecedent and has a similar level of confidence. The characteristics in the antecedent must be the same for the rule to be redundant. In the frequent association rules above, there are 8 redundant rules in the subscribed=yes list, and 7 redundant rules in the subscribed=no list. Non-redundant rules are highlighted green.

In the table below, the reason for rules being redundant is shown by their subsequent non-redundant rule. Each of the redundant rules share one or more antecedents with the non-redundant rule, but the non-redundant rule captures all the common antecedents in one rule.

The minimum confidence is set at 0.8 for this question.

	Subsequent non-redundant rule	Redundant rules
1	default_credit=no past_marketing=success call_duration=500-1k balance=below-1k ==> subscribed=yes	past_marketing=success call_duration=500-1k balance=below-1k ==> subscribed=yes
2	default_credit=no housing=no loan=no past_marketing=success call_duration=500-1k ==> subscribed=yes	housing=no loan=no past_marketing=success call_duration=500-1k ==> subscribed=yes housing=no past_marketing=success call_duration=500-1k ==> subscribed=yes default_credit=no housing=no past_marketing=success call_duration=500-1k ==> subscribed=yes loan=no past_marketing=success call_duration=500-1k ==> subscribed=yes default_credit=no loan=no past_marketing=success call_duration=500-1k ==> subscribed=yes past_marketing=success call_duration=500-1k ==> subscribed=yes default_credit=no past_marketing=success call_duration=500-1k ==> subscribed=yes
3	marital=married ==> subscribed=no	
4	default_credit=no call_duration=100-500s ==> subscribed=no	call_duration=100-500s ==> subscribed=no
5	default_credit=no loan=no past_marketing=unknown ==> subscribed=no	loan=no past_marketing=unknown ==> subscribed=no default_credit=no loan=no ==> subscribed=no default_credit=no past_marketing=unknown ==> subscribed=no loan=no ==> subscribed=no past_marketing=unknown ==> subscribed=no default_credit=no ==> subscribed=no

Task 2: Classification in Weka and Java

A company has created a large medical dataset focusing on COVID-19 infection by collecting people's responses to an online calculator called "Nexoid COVID survival calculator". This calculator generates two risk values based on a person's response to the questions asked by the calculator.

Classification in Weka

Question 1

Choose four classification algorithms from NaiveBayes, IBk, PART, OneR, ZeroR and J48 based on their classification accuracy performance.

Algorithm	**Classification accuracy** *(high)*	**Classification accuracy** *(low)*	**Classification accuracy** *(weighted average)*
NaiveBayes	86.9%	97.3%	94.23%
IBk	87.2%	96.4%	93.67%
PART	87.0%	99.3%	95.64%
OneR	84.7%	99.8%	95.29%
ZeroR	0.0%	100.0%	70.27%
J48	87.2%	99.5%	95.86%

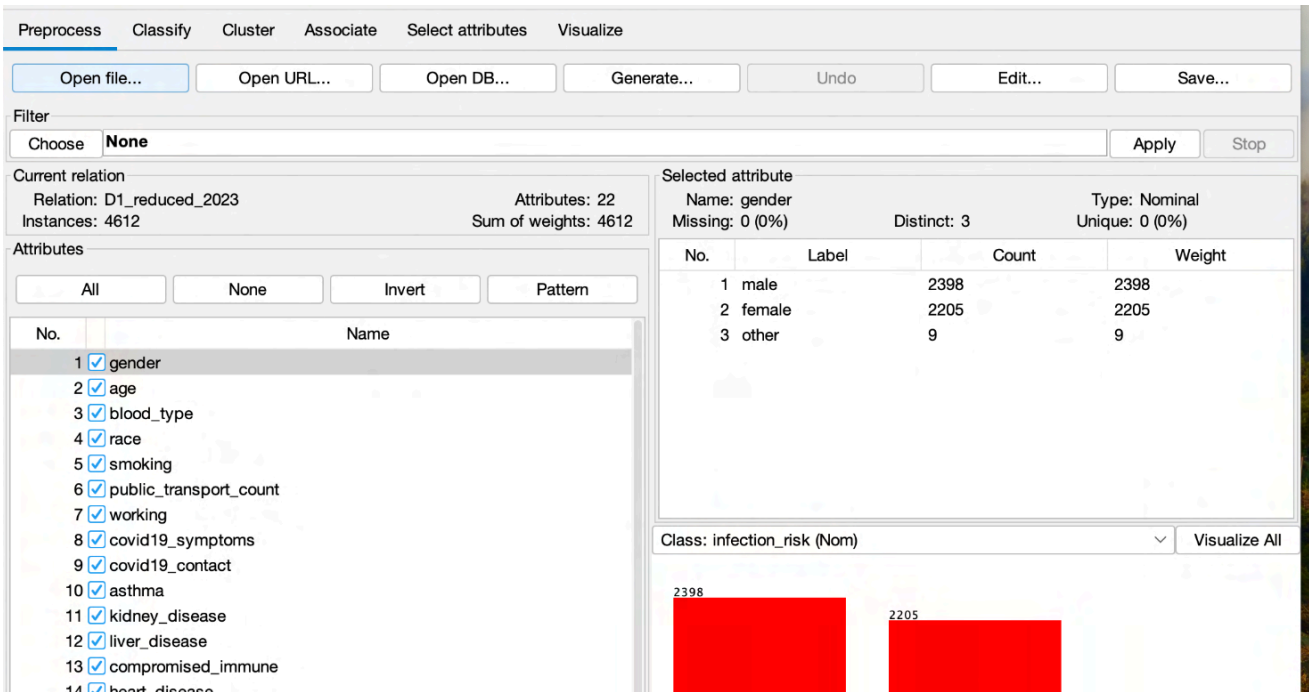
Justification of decision

Selecting four algorithms from the initial six, required an understanding of both the 'weighted average' classification accuracy. The four algorithms chosen have the highest classification accuracy. J48, PART, OneR and NaiveBayes have accuracy above 94%. It's important to note that of the remaining algorithms, IBk was ranked fifth by weighted average classification accuracy, but is the equal best at correctly classifying 'high risk' individuals.

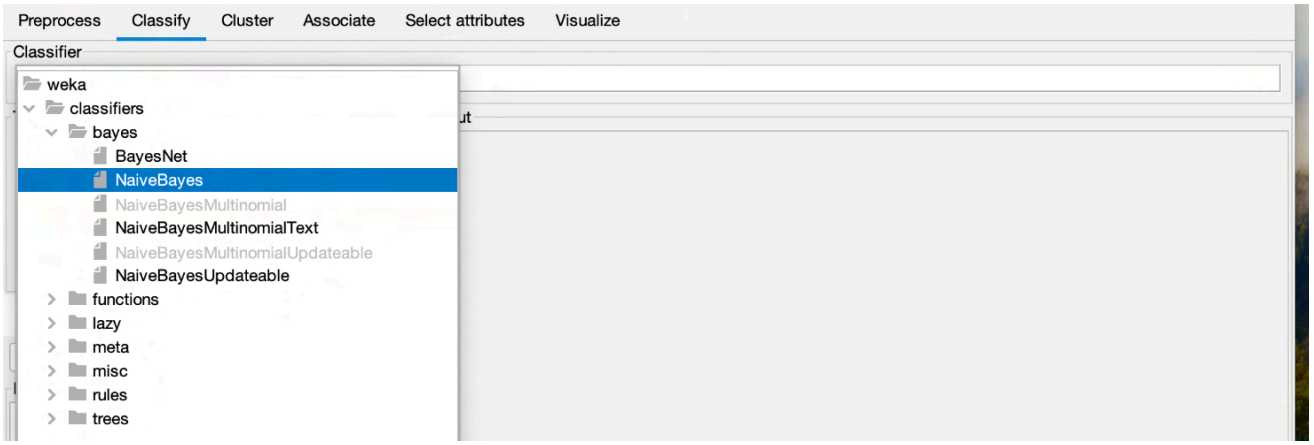
The algorithms chosen after comparison of accuracy are J48, PART, OneR and NaiveBayes.

Description of working process

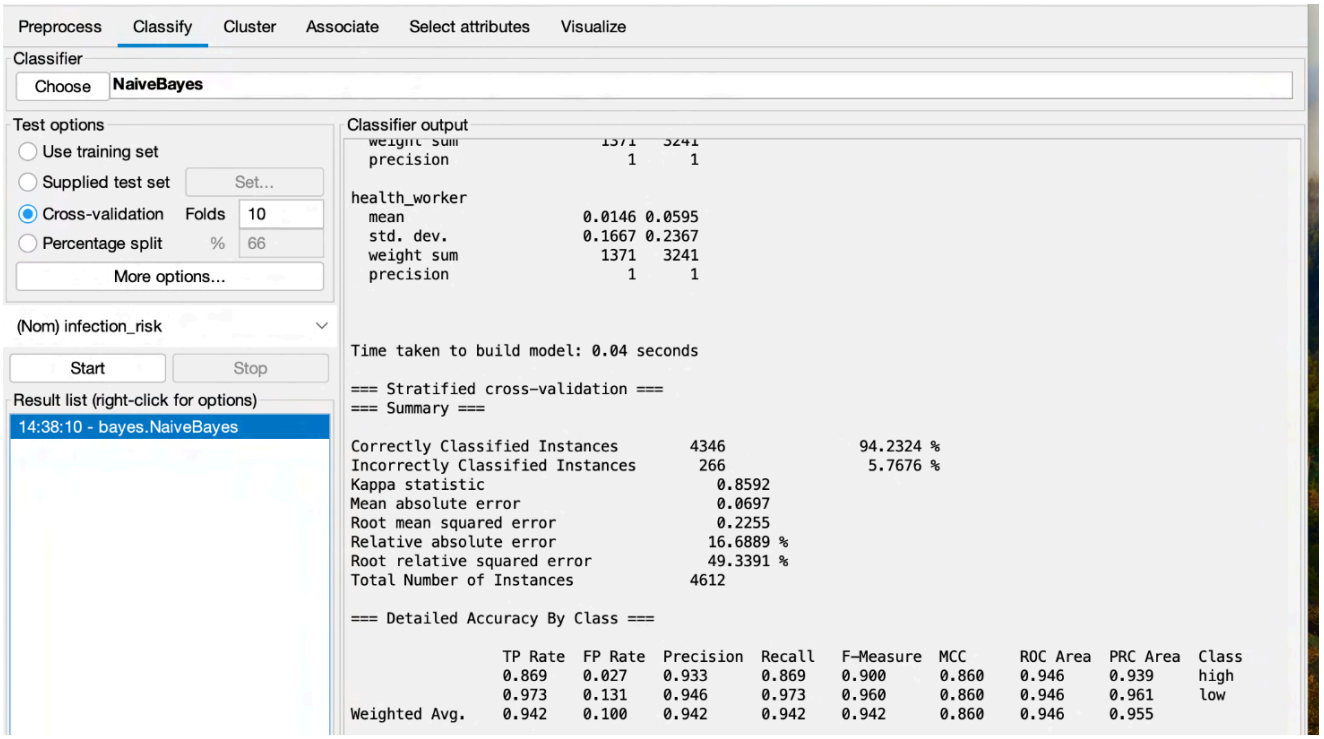
Step 1: In the Preprocess section, open the COVID19.arff file.



Step 2: In the Classify section, select the algorithm as the classifier.



Step 3: Ensure the test option is Cross-validation and Folds=10, and (Nom) infection_risk is selected. After hitting start, note the percentage of Correctly Classified Instances.



Question 2

Choose one evaluator from InfoGainAttributeEval or GainRatioAttributeEval and determine the number of attributes to use for each of the four chosen algorithms in order to achieve a relatively better classification performance by that algorithm. Based on the best classification performance among the four algorithms using the chosen number of attributes, choose two of the four algorithms. What are the two algorithms? Provide the performance comparison.

Number of attributes	Classification accuracy for each algorithm			
	J48	PART	OneR	NaiveBayes
3	95.29%	95.29%	95.29%	94.84%
4	95.29%	95.29%	95.29%	93.58%
5	95.29%	95.32%	95.29%	93.58%
6	95.56%	95.60%	95.29%	93.56%
7	95.71%	95.71%	95.29%	93.54%
8	95.71%	95.71%	95.29%	93.84%
9	95.66%	95.58%	95.29%	94.25%
10	95.95%	95.90%	95.29%	94.26%

Justification of decision

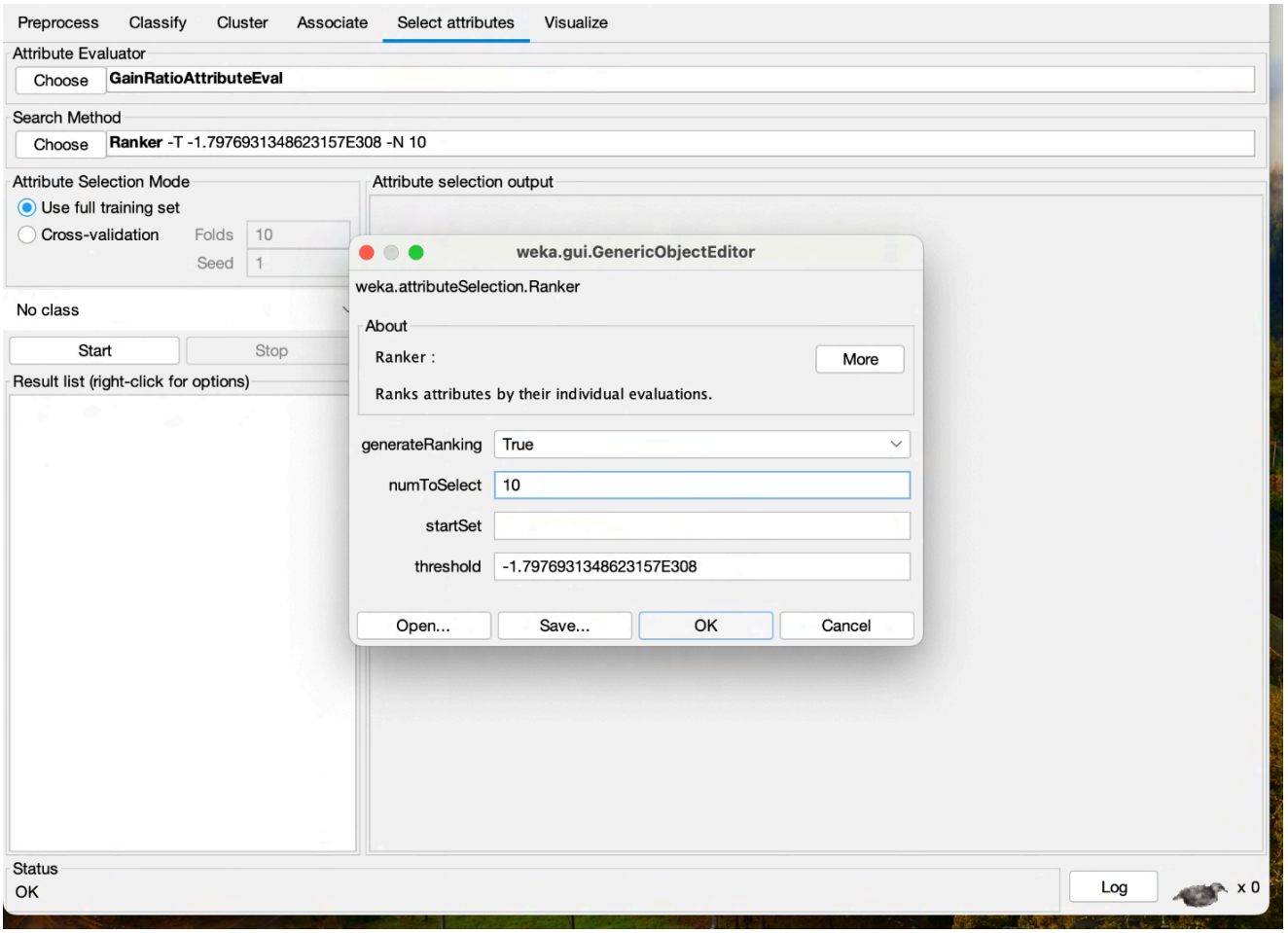
Chosen evaluator for this question is GainRatioAttributeEval, with Ranker search method. Ranked attributes (first to tenth) are: public_transport_count, race, covid19_symptoms, covid19_contact, hiv_positive, nursing_home, kidney_diease, health_worker, working, other_chronic.

Using the Ranker search method for the GainRatioAttributeEval evaluator, each of the four algorithms were tested at three to 10 attributes to find the point at which classification accuracy decreases, or stops improving. For OneR and NaiveBayes, this was at three attributes, and for J48 and PART, it was at seven attributes. Although the performance for both J48 and PART peaks at 10 attributes, there are only 21 attributes in the dataset (plus infection_risk), so performance is expected to improve with every additional attribute. At seven attributes (public_transport_count, race, covid19_symptoms, covid19_contact, hiv_positive, nursing_home and kidney_diease), J48 and PART algorithms have identical classification accuracy at 95.71%.

The algorithms chosen after comparison of accuracy are J48 and PART with seven attributes.

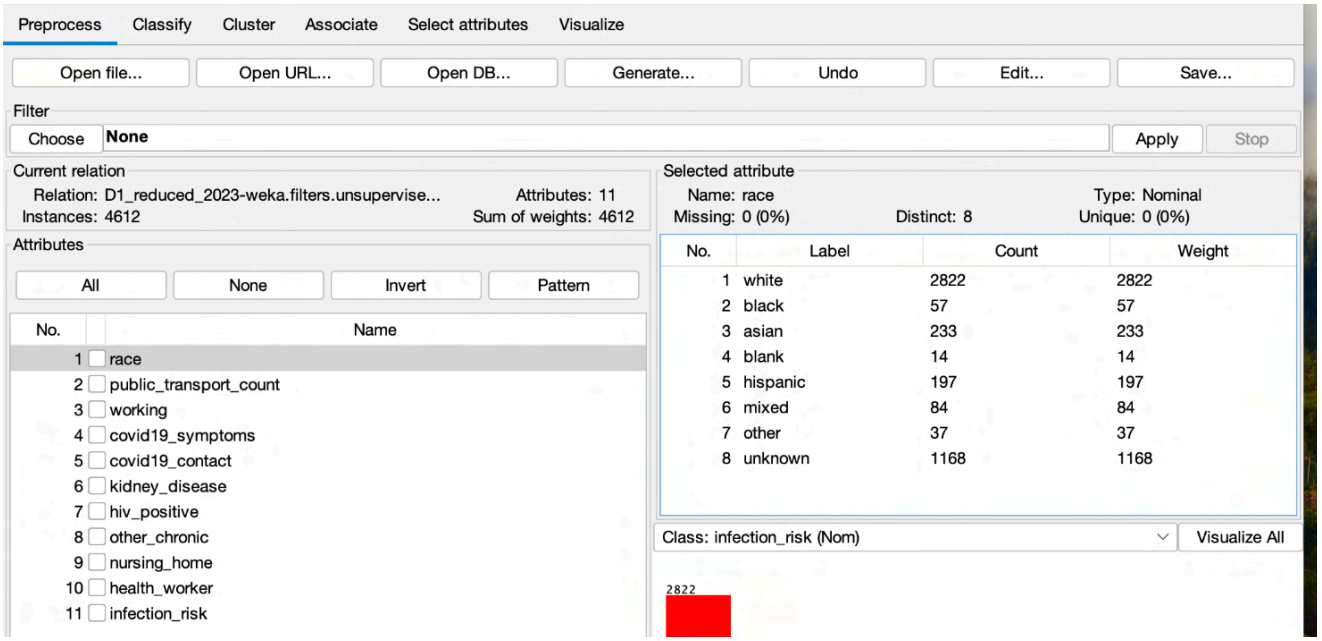
Description of working process

Step 1: After selecting the COVID19.arff file, proceed to the Select Attributes tab. Choose GainRatioAttributeEval as the Attribute Evaluator and Ranker as the Search Method. Ensure numToSelect=10 in the Ranker options to receive only the top 10 ranked attributes in the full training set.

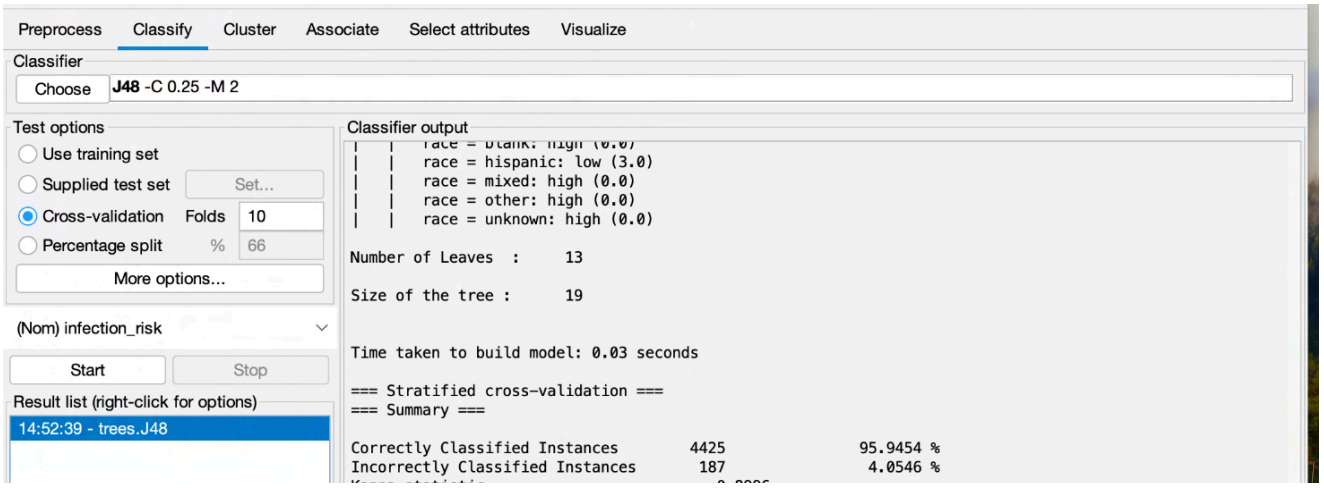


Step 2: Ensure (Nom) infection_risk is selected, and note the 10 highest ranked attributes, in order, after selecting Start.

Step 3: Return to the Preprocess section, and remove attributes that are not ranked in the top 10 or the classification attribute (infection_risk in this case).



Step 4: In the Classify section, choose the appropriate classifier and select Start. Note the percentage of Correctly Classified Instances for the appropriate classifier/algorithm and number of selected attributes.



Step 5: Repeat Step 4, to get the percentage of Correctly Classified Instances for each classifier/algorithm.

Step 6: Return to Step 3 and remove the lowest ranked remaining attribute, before repeating Step 4 and 5 to get the Correctly Classified Instances for each classifier/algorithm. Continue this process until you have the classification accuracy for all classifiers/algorithms from 10 to three selected attributes.

Question 3

Complete a cost sensitive analysis of the two chosen algorithms from Question 2. You can assume that class infection_risk=high is more important to you and you want to minimise the classification error to this class. Based on your analysis, which algorithm would you use in order to minimise the cost? Show your cost matrix and justify your decision by providing evidence.

Algorithm	Base	Cost sensitive classifier minimizeExpectedCost=False	Cost sensitive classifier minimizeExpectedCost=True
PART	198 (91.71%)	892 (95.14%)	920 (95.40%)
J48	198 (91.71%)	890 (95.01%)	918 (95.45%)

Justification of decision

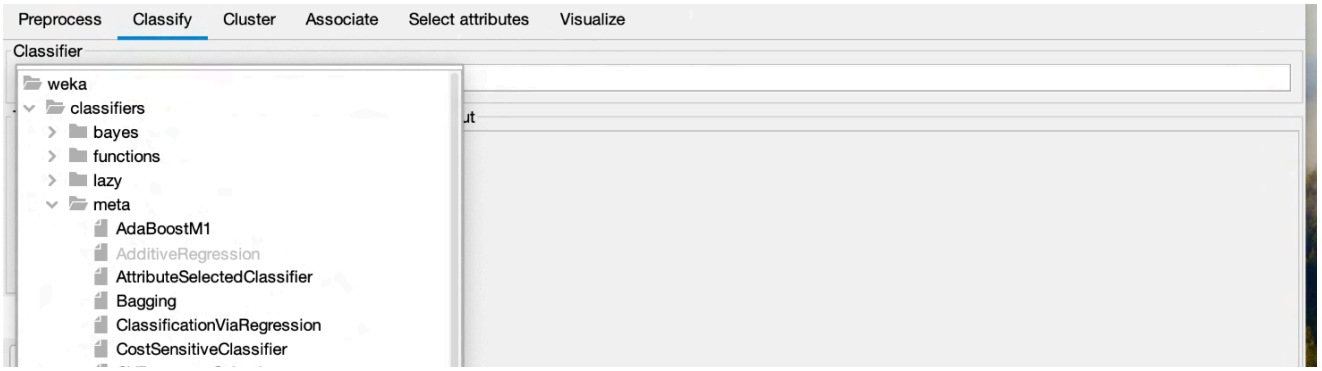
Cost matrix	Predicted class = high	Predicted class = low
Actual class = high	0.0	5.0
Actual class = low	1.0	0.0

The instance of a false negative, where a patient is classified as low risk, but is actually high risk, presents the greatest cost to the company. If a patient is classified as high risk and is actually low risk (false positive), they may seek treatment from medical professionals and be cleared of significant risk to COVID-19 infection. This type of error, while wrong, is inconvenient to the patient's time, but not their health. A false negative is represented by a cost of 5.0 in the cost matrix. A patient classified as low risk, may then engage in normal activities, not knowing they are at a high risk to COVID-19 infection. These individuals face health risks due to the error of the calculator. The matrix is constructed to show the significance of this error in an effort to minimise this type of error.

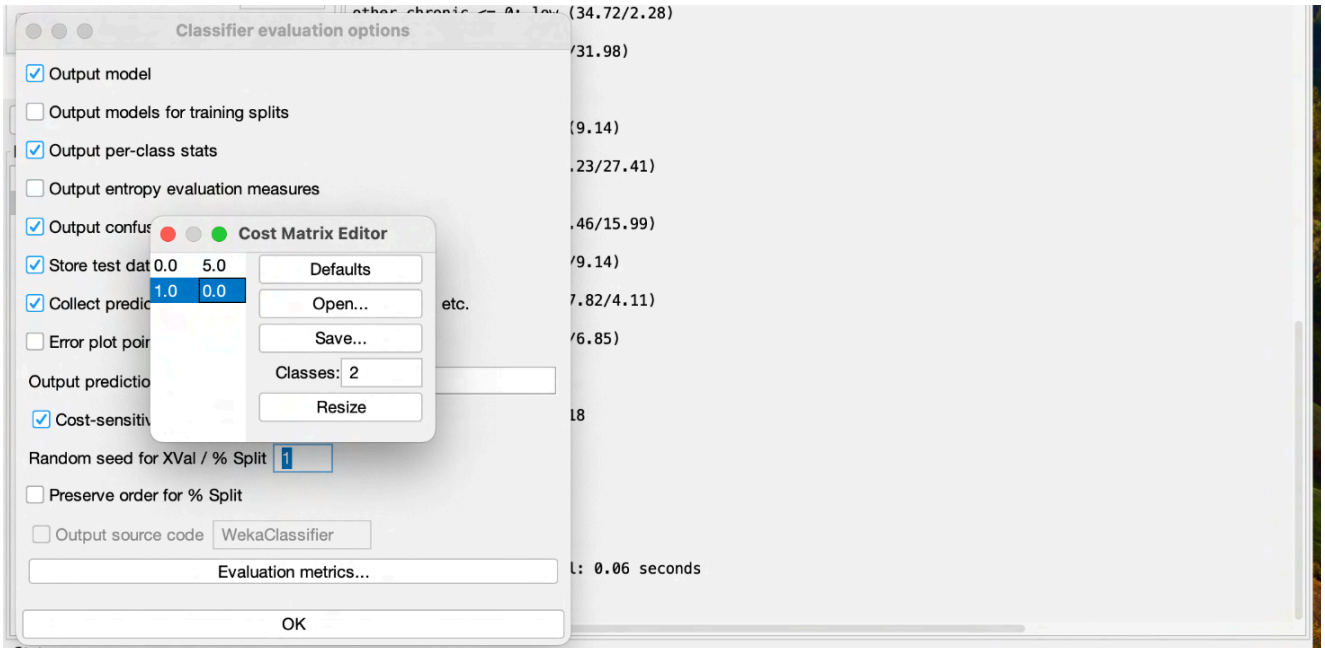
The algorithm chosen to minimise cost is J48. While initial performance and cost is identical to the PART algorithm, J48 has a lower cost when applying the cost matrix. Classification accuracy is best when minimizeExpectedCost=True, and in each of the three cost scenarios, J48 returns the lowest value for individuals predicted as low risk, but actually high risk.

Description of working process

Step 1: In the Preprocess section, select the COVID19.arff, and remove unwanted attributes. Then proceed to the Classify section and choose CostSensitiveClassifier.

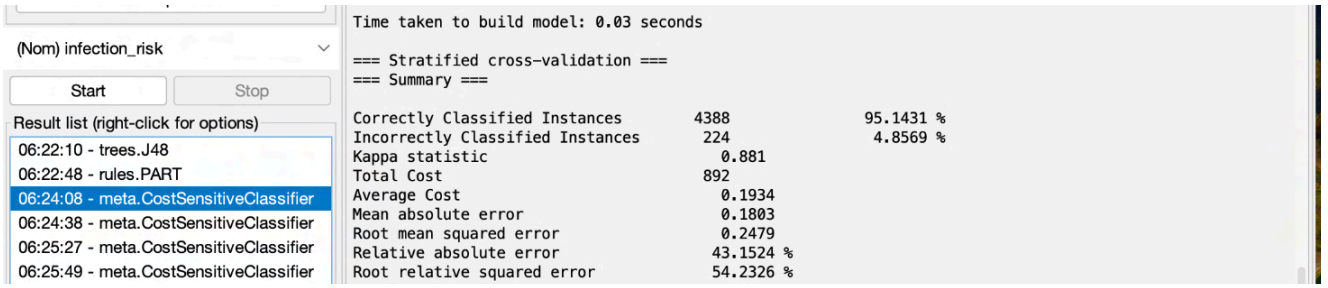


Step 2: Within the More options, select Cost-sensitive evaluation, and set the parameters to match the Cost Matrix.

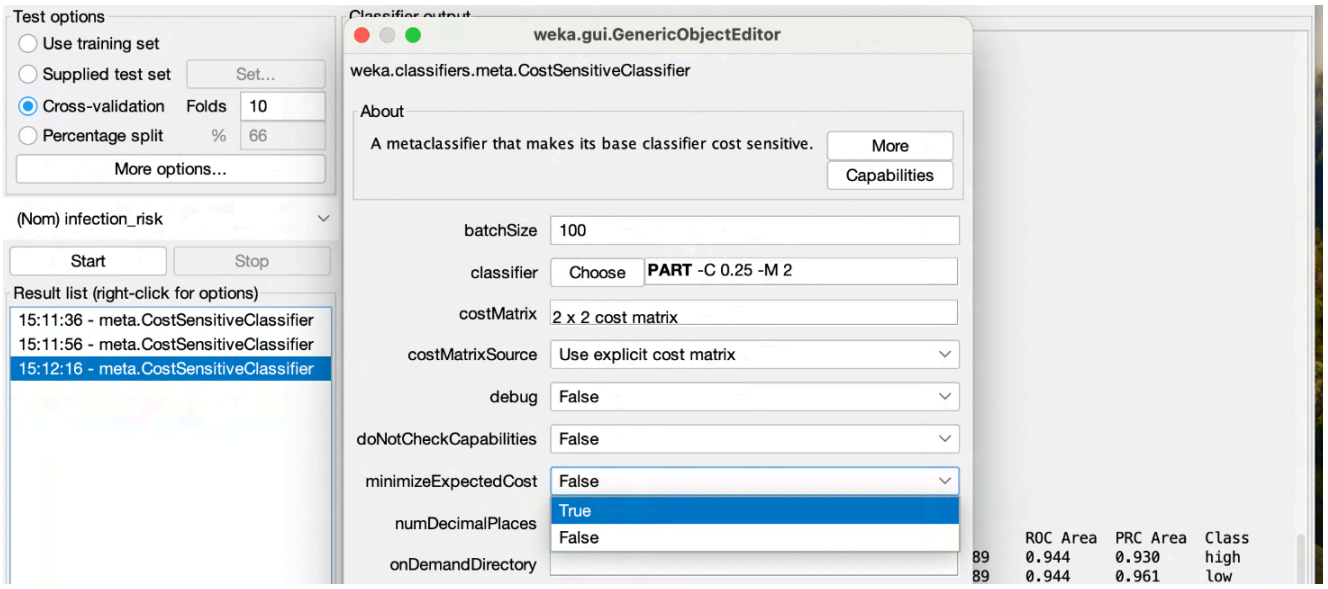


Step 3: Ensure the CostSensitiveClassifer parameters are set correctly. Choose the appropriate classifier, then set the costMatrix to Classes=2 and have values to match your Cost Matrix.

Step 4: Select start and note the Total Cost value, this is added to the Cost sensitive classifier - minimizeExpectedCost=False column.



Step 5: Change the parameters of the CostSensitiveClassifier to minimizeExpectedCost=True.

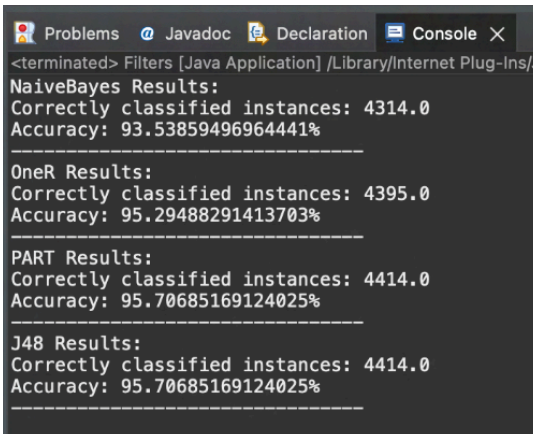


Step 6: Repeat Step 4 and 5 for other classifier/algorithm.

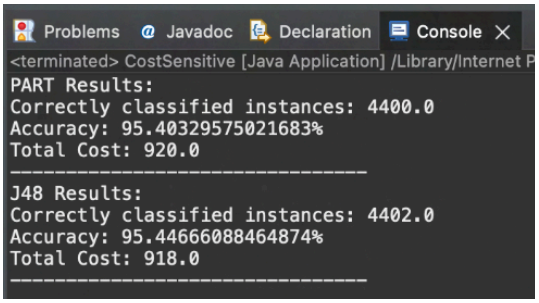
Java Program

Develop a Java program to build classifiers for COVID-19 infection risk prediction. Your program should satisfy the following requirements.

1. *Perform the classification task using the four algorithms chosen in Question 1 of this task with the number of attributes determined in Question 2. Your program should display the correctly classified instances values and accuracy. The accuracy results should be the same as the results obtained in Question 2 using Weka.*



2. *Perform the classification task using the two algorithms chosen in Question 3 of this task by taking cost into consideration. Your program should display classification accuracy and total cost for each classifier. Your program should produce the same results obtained in Question 3 using Weka.*



Task 3: Text classification in Weka and Java

News documents are categorised into four classes: computer, politics, science and sports. Filter the data to select attributes from the documents. Choose 100 attributes and use J48 to do the classification. For the parameters in the filter, tune parameters that are important for determining the attributes.

Attribute selection in Weka

Question 1

Which parameters in the filter do you want to tune? What are the chosen values for these parameters?

	Tuned parameters	**Chosen parameter values**
1	IDFTransform	True
2	TFTransform	True
3	outputWordCounts	True
4	stemmer	LovinsStemmer
5	stopwordsHandler	Rainbow
6	tokenizer	AlphabeticTokenizer

Question 2

Briefly describe your working process in Weka to determine the values for the parameters in the filter. Provide evidence to show your working.

Algorithm	**Classification accuracy**	**Correctly classified instances**	**Time taken to build model**	**Parameter tuned**
J48	58.28%	8,169	14.10s	None
	76.84%	10,771	27.81s	stemmer, stopwordsHandler, IDFTransform and TFTransform
	75.93%	10,644	29.72s	stemmer, stopwordsHandler, IDFTransform, TFTransform, normalizeDocLength and outputWordCounts
	76.93%	10,784	29.11s	stemmer, stopwordsHandler, IDFTransform, TFTransform and outputWordCounts
	77.78%	10,903	29.35s	stemmer, stopwordsHandler, IDFTransform, TFTransform, outputWordCounts and tokenizer

In Question 1, seven parameters were chosen to tune, to measure the impact they have on classification performance. The aim was to determine a set of parameters that would have high

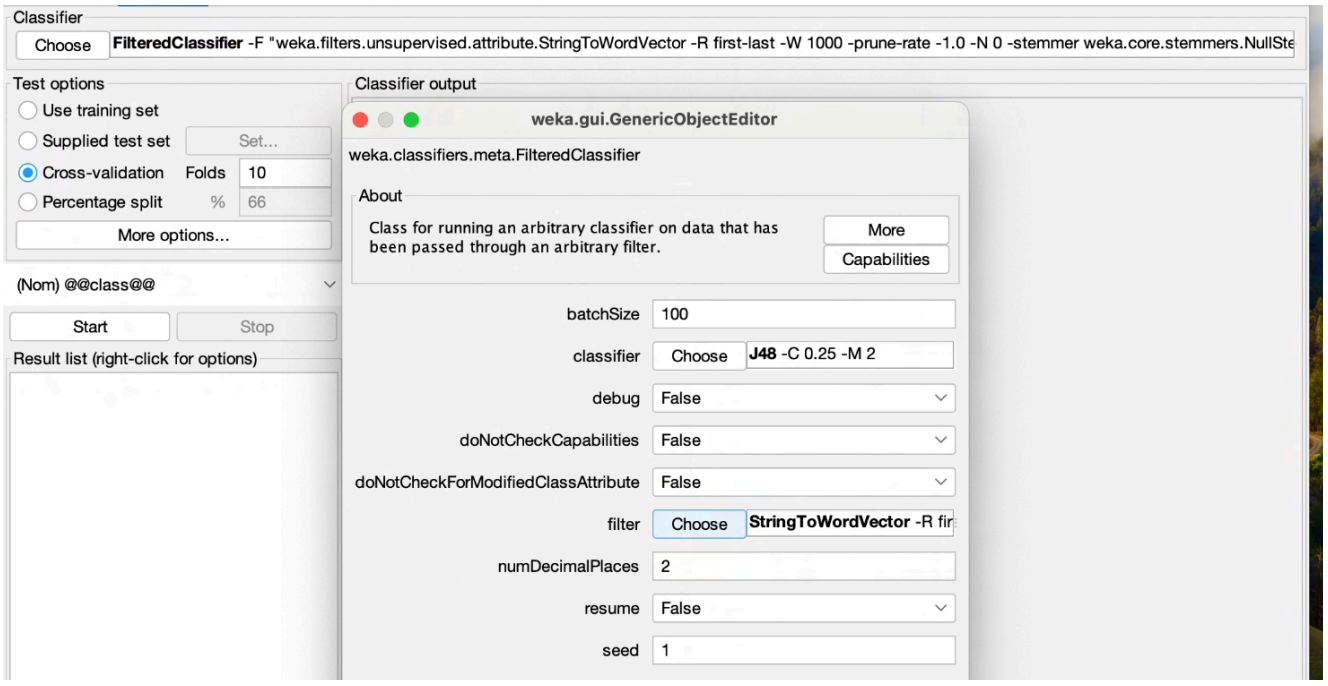
classification accuracy and be used for the Java program Question 1 and 2. It was determined that stemmer, stopwordsHandler, IDFTransform, TFTransform, outputWordCounts and tokenizer are the parameters to be tuned from default settings. This combination provided optimal classification accuracy. The parameter, normalizeDocLength was used, but it did not improve performance and hindered the time efficiency.

As seen above, tuning filter parameters can significantly improve the accuracy of an algorithm by enhancing feature extraction and representation. Setting IDFTransform to True applies Inverse Document Frequency (IDF) scaling, which reduces the impact of common words across documents, allowing the model to give more importance to unique terms. TFTransform to True normalises term frequency, giving a nuanced representation of word significance within documents to help distinguish between classes. Enabling outputWordCounts to True ensures that the occurrences for each term is considered, rather than just the presence or absence. Using the LovinsStemmer reduces words to their root forms, which groups similar terms together, effectively reducing unnecessary noise. The Rainbow stopWordsHandler filters out common words (stop words) that do not contribute meaningfully to classification, thus focusing the model on key terms and the AlphabeticTokenizer extracts only alphabetic tokens, excluding punctuation and numbers, which can introduce noise, thereby refining the text features used. These settings improve the algorithm's ability to accurately predict class labels of either computer, politics, science or sport.

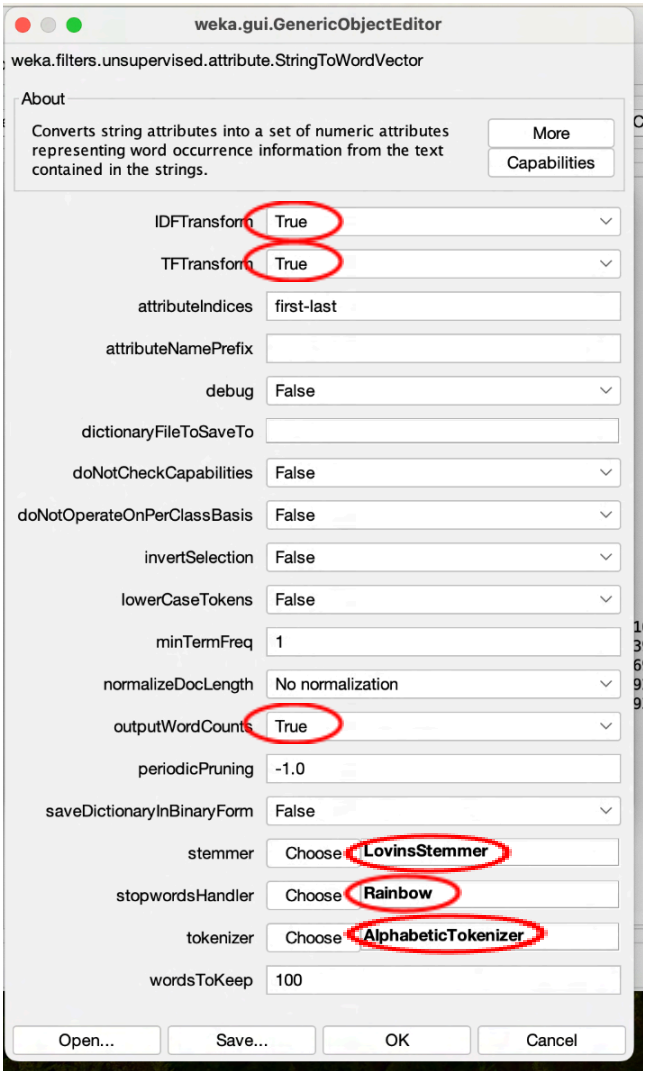
Screenshots below provide evidence of Weka classification and parameter tuning.

Step 1: Select the news.arff file in the Preprocess section.

Step 2: Select FilteredClassifier in the Classify section, and select StringToWordVector as the filter.



Step 3: Modify the parameters within the StringToWordVector settings, click start, and note the time taken to run the model, correctly classified instances and classification accuracy.



Java Program

Question 1

Perform the classification task using four classification algorithms, IBk, SMO, J48 and the method HoeffdingTree in Weka, and use a filter with the parameter settings determined in Question 1 of this task.

Weka classification data table

Algorithm	Classification accuracy	Correctly classified instances	Time taken to build model
IBk	73.60%	10,317	2.52s
SMO	83.22%	11,666	9.53s
J48	77.78%	10,903	29.35s
HoeffdingTree	72.69%	10,190	4.48s

Question 2

Develop a Java program to classify the documents in the news dataset. The program should display the correctly classified instances results, accuracy, and the time taken by each algorithm.

Java classification output image

```
===== IBk =====
Time taken: 5067 ms
Correctly Classified Instances: 10317.0/14018
Accuracy: 73.59823084605507%

Correctly Classified Instances       10317               73.5982 %
Incorrectly Classified Instances      3701               26.4018 %
Kappa statistic                          0.6379
Mean absolute error                      0.1332
Root mean squared error                  0.3616
Relative absolute error                 36.4373 %
Root relative squared error             84.5882 %
Total Number of Instances            14018

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.773    0.135    0.753      0.773    0.763      0.634    0.816     0.673     computer
                 0.747    0.060    0.743      0.747    0.745      0.686    0.845     0.608     politics
                 0.689    0.124    0.685      0.689    0.687      0.564    0.782     0.571     science
                 0.726    0.047    0.776      0.726    0.750      0.697    0.840     0.623     sports
Weighted Avg.    0.736    0.102    0.736      0.736    0.736      0.636    0.816     0.623

===== Evaluation done =====

===== SMO =====
Time taken: 11500 ms
Correctly Classified Instances: 11663.0/14018
Accuracy: 83.20017120844628%

Correctly Classified Instances       11663               83.2002 %
Incorrectly Classified Instances      2355               16.7998 %
Kappa statistic                          0.7683
Mean absolute error                      0.2689
Root mean squared error                  0.3404
Relative absolute error                 73.5759 %
Root relative squared error             79.6266 %
Total Number of Instances            14018

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.902    0.104    0.822      0.902    0.860      0.782    0.928     0.804     computer
                 0.813    0.024    0.888      0.813    0.849      0.817    0.933     0.803     politics
                 0.766    0.094    0.761      0.766    0.763      0.670    0.851     0.657     science
                 0.819    0.016    0.920      0.819    0.867      0.841    0.942     0.830     sports
Weighted Avg.    0.832    0.070    0.835      0.832    0.832      0.768    0.910     0.767

===== Evaluation done =====
```

```
===== J48 =====
Time taken: 33277 ms
Correctly Classified Instances: 10903.0/14018
Accuracy: 77.77857040947353%

Correctly Classified Instances       10903               77.7786 %
Incorrectly Classified Instances      3115               22.2214 %
Kappa statistic                          0.6957
Mean absolute error                      0.1267
Root mean squared error                  0.3097
Relative absolute error                 34.6803 %
Root relative squared error             72.4383 %
Total Number of Instances            14018

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.822    0.104    0.809      0.822    0.816      0.716    0.886     0.772     computer
                 0.788    0.049    0.788      0.788    0.788      0.739    0.885     0.719     politics
                 0.688    0.120    0.693      0.688    0.690      0.569    0.809     0.632     science
                 0.821    0.035    0.838      0.821    0.830      0.792    0.911     0.792     sports
Weighted Avg.    0.778    0.085    0.778      0.778    0.778      0.693    0.869     0.726

===== Evaluation done =====

===== HoeffdingTree =====
Time taken: 15025 ms
Correctly Classified Instances: 10190.0/14018
Accuracy: 72.69225281780568%

Correctly Classified Instances       10190               72.6923 %
Incorrectly Classified Instances      3828               27.3077 %
Kappa statistic                          0.6205
Mean absolute error                      0.1366
Root mean squared error                  0.367
Relative absolute error                 37.3866 %
Root relative squared error             85.8485 %
Total Number of Instances            14018

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.932    0.240    0.675      0.932    0.783      0.659    0.937     0.849     computer
                 0.664    0.026    0.853      0.664    0.747      0.705    0.946     0.806     politics
                 0.452    0.069    0.720      0.452    0.556      0.452    0.864     0.687     science
                 0.824    0.056    0.768      0.824    0.795      0.748    0.953     0.866     sports
Weighted Avg.    0.727    0.118    0.738      0.727    0.714      0.626    0.921     0.798

===== Evaluation done =====
```

Question 3

Which classifier performs the best in terms of time efficiency? Describe why this algorithm is faster than others.

In both Weka and Java, IBk is the classifier that performs best in terms of time efficiency. Time outputs show that IBk is quicker than HoeffdingTree and SMO, and significantly faster than J48. While the SMO classifier produces the highest classification accuracy, IBk is the most efficient. IBk is a lazy k-Nearest Neighbour learning algorithm, which means it doesn't perform heavy computation while training.

The IBk algorithm doesn't build a complex or computationally intensive model building during training. It simply stores the training instances and performs the classification during prediction. The trade off for an efficient training model is that the prediction phase can be slower. The time taken to train the model is faster than SMO, J48 and Hoeffding Tree as IBk doesn't generalise the training data, but instead keeps all the data in memory, thus skipping the steps of analysing and fitting the data into a predictive model.

The SMO algorithm involves the solving of complex optimisation problems which is computationally expensive and time consuming, especially with high-dimensional data. J48 and HoeffdingTree algorithms need to build and refine tree structures which involve selecting the best splits, evaluating multiple attributes and pruning. Time to monitor inputs (HoeffdingTree) and searching multiple attributes (J48), takes time and computational resources.

IBk is faster in comparison to these other algorithms because it does not perform any model-building steps; it simply stores the data.