IFN645: Large Scale Data Mining

**Capstone Project**

Group 17

Yuen Man Ho, N11449381

Wing Hei Siu Samuel, N11423579

**Table of Contents**

Statement of completeness…………………………………………………………………..3

Task 1: Association mining in Java (12 marks)……………………………………………4

Task 2: Classification in Weka and Java (12 marks)…………………………………… 11

Task 3: Text classification in Weka and Java (10 marks)……………………………….30

## **Statement of completeness**

|  |  |  |
| --- | --- | --- |
| **Statement of completeness:**  **Group Number: 17** | | |
| The following undersigned members of the group agree to abide by this statement to ensure successful completion of the project (Assignment 2) to meet project requirements and timelines. We declare that each team member has a same or similar contribution to the project. | | |
| **Name, student number & email** | **Signature** | **Date** |
| ***Yuen Man Ho, N11449381, n11449381@qut.edu.au*** | *Fiona* | *16/10/2024* |
| **Wing Hei Siu, N11423579, *N11423579@qut.edu.au*** | *Samuel* | *16/10/2024* |
|  |  |  |
| Only if permitted |  |  |
| ***Task Allocation for each student****Yuen Man Ho, Q1, Q3.1* *Wing Hei Siu, Q2, Q3.2* | ***Allocation Percentage* (%)**50%50% | |
| ***Other issues or comments*** | | |

## 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 to their customers.

Questions

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

A graph showing the growth of a company

Description automatically generated**Answer:**

|  |  |  |  |
| --- | --- | --- | --- |
| Total Time (ms) | Algorithm | | |
| Min Support | **Frequent Itemsets**  **Count** | **Apriori** | **FP-Growth** |
| 0.1 | 1482 | 1762 | 156 |
| 0.2 | 401 | 697 | 76 |
| 0.3 | 154 | 136 | 62 |
| 0.4 | 81 | 73 | 49 |
| 0.5 | 37 | 52 | 64 |
| 0.6 | 18 | 32 | 59 |
| 0.7 | **11** | 42 | 41 |
| 0.8 | **7** | 23 | 39 |
| 0.9 | **1** | 23 | 33 |
| Grand Total |  | **2840** | **579** |

Based on the above chart, below is the analysis of Apriori and FP-Growth algorithms' performance in terms of time efficiency:

1. **Comparison of Time Efficiency**:
   * Apriori algorithm initially takes much more time to generate frequent patterns compared to FP-Growth when the minimum support is low (e.g., at 0.1 and 0.2). This is evident from the steep drop in execution time for Apriori, starting at 1762 ms and reducing to 697 ms, while FP-Growth maintains a relatively stable execution time of around 156 ms.
2. **Turning Point at Minimum Support 0.4**:
   * As the minimum support increases, the time difference between the two algorithms decreases. At a minimum support of 0.4, Apriori's execution time (49 ms) becomes comparable to FP-Growth’s (94 ms), indicating that both algorithms are performing similarly.
3. **After Minimum support 0.4**:
   * Apriori becomes more efficient, taking less time than FP-Growth. This trend continues up to a minimum support of 0.9, where Apriori consistently takes less time to compute frequent patterns.
4. **Observations on Frequent Itemset Count**:
   * Another key insight from the chart is that as the minimum support increases, the number of frequent itemsets found decreases significantly. This reduction in itemsets directly impacts the execution time, particularly for Apriori, which shows dramatic time reductions as the frequent itemset count reduces.

**Conclusion:**

* **For Low Minimum Support Values** (0.1 and 0.2), FP-Growth is the more time-efficient algorithm, significantly outperforming Apriori.
* **For Medium to High Minimum Support Values** (0.4 and above), Apriori becomes more efficient, taking less time than FP-Growth.

1. Choose a frequent pattern mining algorithm based on the comparison in 1), state the reason to choose this algorithm. Then use the chosen algorithm to generate the top 5 most frequent size-3 patterns from the yes-class dataset and 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.

**Answer:**

**Algorithm Selection:**

* **Chosen Algorithm:** FP-Growth
* **Reason for Choice:** FP-Growth was chosen over Apriori due to time efficiency concluded above, when minimum support threshold of 0.3. While Apriori requires multiple database scans and generates numerous candidate patterns, FP-Growth uses a compact data structure (FP-tree) to store frequent patterns, reducing the need for redundant scans and computational time.

**Minimum Support:**

* **Minimum Support Value: Yes= 0.45 and No=0.48**
* This support threshold ensures that only top 5 most frequent size-3 patterns from yes-class and no-class dataset are generated.

Java output:

Frequent patterns for 'yes' dataset:

============= FP-GROWTH 2.42 - STATS =============

Transactions count from database : 5289

Max memory usage: 133.56075286865234 mb

Frequent itemsets count : 5

Total time ~ 105 ms

===================================================

Top 5 most frequent size-3 patterns for 'yes' class:

Top 5 size-3 patterns:

5 7 8 #SUP: 3120

5 8 29 #SUP: 2988

5 8 9 #SUP: 2712

1 5 8 #SUP: 2519

3 5 8 #SUP: 2471

Frequent patterns for 'no' dataset:

============= FP-GROWTH 2.42 - STATS =============

Transactions count from database : 39922

Max memory usage: 133.56075286865234 mb

Frequent itemsets count : 5

Total time ~ 164 ms

===================================================

Top 5 most frequent size-3 patterns for 'no' class:

Top 5 size-3 patterns:

5 8 12 #SUP: 27371

5 9 12 #SUP: 21352

5 8 9 #SUP: 21217

5 12 16 #SUP: 20357

5 8 16 #SUP: 19799

“Yes” dataset

Below are the top 5 most frequent size-3 patterns for 'yes' class:

**Total Transactions:** 5,289

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Pattern | Attributes | Support |
| Yes | 5 7 8 | Default Credit = No, Housing = No, Loan = No | 3,120 |
| Yes | 5 8 29 | Default Credit = No, Loan = No, Call Duration = 500-1k seconds | 2,988 |
| Yes | 5 8 9 | Default Credit = No, Loan = No, Call Duration = 100-500 seconds | 2,712 |
| Yes | 1 5 8 | Age = 20-below, Default Credit = No, Loan = No | 2,519 |
| Yes | 3 5 8 | Marital = Single, Default Credit = No, Loan = No | 2,471 |

“No” dataset

Below are the top 5 most frequent size-3 patterns for 'no’ class:

**Total Transactions:** 39,922

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Pattern | Attributes | Support |
| No | 5 8 12 | Default Credit = No, Loan = No, Call Duration = 100s-below | 27,371 |
| No | 5 9 12 | Default Credit = No, Call Duration = 100-500s, Call Duration = 100s-below | 21,352 |
| No | 5 8 9 | Default Credit = No, Loan = No, Call Duration = 100-500 seconds | 21,217 |
| No | 5 12 16 | Default Credit = No, Call Duration = 100s-below, Job = Management | 20,357 |
| No | 5 8 16 | Default Credit = No, Loan = No, Job = Management | 19,799 |

**Common Characteristics:**

* Both 'yes' and 'no' classes show that the attribute **Default Credit = No** is a main feature in most frequent patterns, indicating that customers without credit defaults are important across both classes.
* Another common characteristic is **Loan = No**. Most top patterns from both classes include customers without personal loans, highlighting that this characteristic is shared among customers regardless of subscription selection.

**Different Characteristics:**

* **Call Duration:**
  + In the 'yes' class, top patterns include customers with longer call durations, such as **500-1k seconds**.
  + In the 'no' class, shorter call durations (**below 100 seconds**) are more important, suggesting that longer call durations may be more indicative of a successful subscription.
* **Demographics:**
  + For the 'yes' class, younger customers (e.g., **Age = 20-below**) and those who are **single** are more commonly found in the top patterns.
  + In the 'no' class, specific job categories like **Management** are more frequently observed.

Summary: Some characteristics are found no matter the subscription status, while younger customer and single status are more frequent found in “yes” class.

1. Generate the top 5 most frequent maximum patterns from yes-class and no class datasets separately, identify any similarity or differences 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.

Answer: Java output:

Transactions count from database : 5289

Max memory usage: 15.900062561035156 mb

Maximal frequent itemset count : 5

Total time ~ 124 ms

===================================================

Conversion completed: C:/Users/Fiona/IFN645/Project/output/Q3\_final\_fpMax\_yes.txt

Top 5 patterns:

Pattern: default\_credit=no loan=no housing=no | Support: 3120

Pattern: default\_credit=no loan=no past\_marketing=unknown | Support: 2988

Pattern: default\_credit=no age=21-30s | Support: 2781

Pattern: default\_credit=no marital=married | Support: 2735

Pattern: default\_credit=no loan=no call\_duration=100-500s | Support: 2712

============= FP-Max v0.96r14 - STATS =============

Transactions count from database : 39922

Max memory usage: 85.86056518554688 mb

Maximal frequent itemset count : 5

Total time ~ 167 ms

===================================================

Conversion completed: C:/Users/Fiona/IFN645/Project/output/Q3\_final\_fpMax\_no.txt

Top 5 patterns:

Pattern: default\_credit=no past\_marketing=unknown loan=no | Support: 27371

Pattern: default\_credit=no marital=married | Support: 24031

Pattern: default\_credit=no housing=yes | Support: 22789

Pattern: default\_credit=no past\_marketing=unknown call\_duration=100-500s | Support: 21352

Pattern: default\_credit=no loan=no call\_duration=100-500s | Support: 21217

**Minimum Support Value:**

* **Minimum Support Value: Yes= 0.5 and No=0.53**
* **Yes-Class Patterns (Minimum Support: 0.5)**

Pattern: default\_credit=no, loan=no, housing=no | Support: 3120

Pattern: default\_credit=no, loan=no, past\_marketing=unknown | Support: 2988

Pattern: default\_credit=no, age=21-30s | Support: 2781

Pattern: default\_credit=no, marital=married | Support: 2735

Pattern: default\_credit=no, loan=no, call\_duration=100-500s | Support: 2712

* **No-Class Patterns (Minimum Support: 0.53)**

Pattern: default\_credit=no, past\_marketing=unknown, loan=no | Support: 27371

Pattern: default\_credit=no, marital=married | Support: 24031

Pattern: default\_credit=no, housing=yes | Support: 22789

Pattern: default\_credit=no, past\_marketing=unknown, call\_duration=100-500s | Support: 21352

Pattern: default\_credit=no, loan=no, call\_duration=100-500s | Support: 21217

**Similarities:**

1. Both classes have the pattern ***default\_credit=no combined with loan=no and call\_duration=100-500s,*** which appears as a frequent pattern in both datasets. This suggests that no matter Yes or No class, customers who do not default on credit, had no loan, and had medium call durations are prevalent.
2. The combination of ***default\_credit=no and marital=married*** appears as a frequent pattern in both datasets, indicating that married individuals with no credit default are common in both classes.

**Differences:**

1. **Housing**: In the yes-class, there is no housing as important attribute, while in the no-class, the pattern default\_credit=no, housing=yes has a high support (22789). This indicates that housing status plays a more prominent role in the no-class dataset.
2. **Past Marketing**: In the no-class, combination of past\_marketing=unknown is more frequent and has a higher support (27371 and 21352) compared to the yes-class, where past\_marketing=unknown only appears once with a lower support (2988).
3. **Age Group**: The yes-class contains a pattern that includes the age group 21-30s (support: 2781), whereas no such age-based pattern is found in the top 5 patterns of the no-class.
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 that you have used. Observe the rules and identify any redundant rules in each set of the rules. You can round the confidence value to three decimal places. If there exist redundant rules, list them and state why you think they are redundant.

Answer: Java output:

============= TOP-K CLASS RULES SPMF v.2.28 - STATS =============

Minsup : 978

Maxsup : 2147483647

Rules count: 10

Memory : 101.79830932617188 mb

Total time : 160 ms

===================================================

Top-k rules saved to: C:/Users/Fiona/IFN645/Project/output/Q4\_topk\_rules\_yes.txt

Converted output saved to: C:/Users/Fiona/IFN645/Project/output/Q4\_topk\_rules\_with\_names\_yes.txt

Rules after sorting by confidence:

Rule: default\_credit=no past\_marketing=success ==> subscribed=yes | Support: 978 | Confidence: 0.648

Rule: past\_marketing=success ==> subscribed=yes | Support: 978 | Confidence: 0.647

Rule: loan=no call\_duration=500-1k ==> subscribed=yes | Support: 1448 | Confidence: 0.394

Rule: default\_credit=no loan=no call\_duration=500-1k ==> subscribed=yes | Support: 1428 | Confidence: 0.393

Rule: call\_duration=500-1k ==> subscribed=yes | Support: 1646 | Confidence: 0.380

Rule: default\_credit=no call\_duration=500-1k ==> subscribed=yes | Support: 1614 | Confidence: 0.379

Rule: loan=no past\_marketing=unknown call\_duration=500-1k ==> subscribed=yes | Support: 1079 | Confidence: 0.360

Rule: default\_credit=no loan=no past\_marketing=unknown call\_duration=500-1k ==> subscribed=yes | Support: 1061 | Confidence: 0.359

Rule: past\_marketing=unknown call\_duration=500-1k ==> subscribed=yes | Support: 1242 | Confidence: 0.350

Rule: default\_credit=no past\_marketing=unknown call\_duration=500-1k ==> subscribed=yes | Support: 1214 | Confidence: 0.349

Sorted rules saved to: C:/Users/Fiona/IFN645/Project/output/sorted\_Q4\_topk\_rules\_with\_names\_yes.txt

============= TOP-K CLASS RULES SPMF v.2.28 - STATS =============

Minsup : 24459

Maxsup : 2147483647

Rules count: 10

Memory : 38.55638885498047 mb

Total time : 67 ms

===================================================

Top-k rules saved to: C:/Users/Fiona/IFN645/Project/output/Q4\_topk\_rules\_no.txt

Converted output saved to: C:/Users/Fiona/IFN645/Project/output/Q4\_topk\_rules\_with\_names\_no.txt

Rules after sorting by confidence:

Rule: past\_marketing=unknown ==> subscribed=no | Support: 33573 | Confidence: 0.908

Rule: default\_credit=no past\_marketing=unknown ==> subscribed=no | Support: 32862 | Confidence: 0.908

Rule: loan=no past\_marketing=unknown ==> subscribed=no | Support: 27814 | Confidence: 0.902

Rule: default\_credit=no loan=no past\_marketing=unknown ==> subscribed=no | Support: 27371 | Confidence: 0.902

Rule: call\_duration=100-500s ==> subscribed=no | Support: 26044 | Confidence: 0.900

Rule: default\_credit=no call\_duration=100-500s ==> subscribed=no | Support: 25538 | Confidence: 0.899

Rule: marital=married ==> subscribed=no | Support: 24459 | Confidence: 0.899

Rule: default\_credit=no ==> subscribed=no | Support: 39159 | Confidence: 0.882

Rule: loan=no ==> subscribed=no | Support: 33162 | Confidence: 0.873

Rule: default\_credit=no loan=no ==> subscribed=no | Support: 32685 | Confidence: 0.873

Sorted rules saved to: C:/Users/Fiona/IFN645/Project/output/sorted\_Q4\_topk\_rules\_with\_names\_no.txt

Due to class imbalance, we choose minimum support for Yes class as 0.5 and No class as 0.3.

Below are top 10 most frequent association rules with subscribed=yes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Antecedents** | **Consequent** | **Support** | **Confidence** |
| 1 | default\_credit=no, past\_marketing=success | subscribed=yes | 978 | 0.648 |
| 2 | past\_marketing=success | subscribed=yes | 978 | 0.647 |
| 3 | loan=no, call\_duration=500-1k | subscribed=yes | 1448 | 0.394 |
| 4 | default\_credit=no, loan=no, call\_duration=500-1k | subscribed=yes | 1428 | 0.393 |
| 5 | call\_duration=500-1k | subscribed=yes | 1646 | 0.38 |
| 6 | default\_credit=no, call\_duration=500-1k | subscribed=yes | 1614 | 0.379 |
| 7 | loan=no, past\_marketing=unknown, call\_duration=500-1k | subscribed=yes | 1079 | 0.36 |
| 8 | default\_credit=no, loan=no, past\_marketing=unknown, call\_duration=500-1k | subscribed=yes | 1061 | 0.359 |
| 9 | past\_marketing=unknown, call\_duration=500-1k | subscribed=yes | 1242 | 0.35 |
| 10 | default\_credit=no, past\_marketing=unknown, call\_duration=500-1k | subscribed=yes | 1214 | 0.349 |

Below are top 10 most frequent association rules with subscribed=no:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Antecedents** | **Consequent** | **Support** | **Confidence** |
| 1 | past\_marketing=unknown | subscribed=no | 33573 | 0.908 |
| 2 | default\_credit=no, past\_marketing=unknown | subscribed=no | 32862 | 0.908 |
| 3 | loan=no, past\_marketing=unknown | subscribed=no | 27814 | 0.902 |
| 4 | default\_credit=no, loan=no, past\_marketing=unknown | subscribed=no | 27371 | 0.902 |
| 5 | call\_duration=100-500s | subscribed=no | 26044 | 0.9 |
| 6 | default\_credit=no, call\_duration=100-500s | subscribed=no | 25538 | 0.899 |
| 7 | marital=married | subscribed=no | 24459 | 0.899 |
| 8 | default\_credit=no | subscribed=no | 39159 | 0.882 |
| 9 | loan=no | subscribed=no | 33162 | 0.873 |
| 10 | default\_credit=no, loan=no | subscribed=no | 32685 | 0.873 |

**Redundant Rules (Subscribed=Yes)**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rule 1 | Rule 2 | Confidence (Rule 1 / Rule 2) | Support (Rule 1 / Rule 2) | Redundant Reason |
| past\_marketing=success ⇒ subscribed=yes | default\_credit=no, past\_marketing=success ⇒ subscribed=yes | 0.647 / 0.648 | 978 / 978 | Rule 2 adds *default\_credit=no* but has almost the same confidence as Rule 1. |

**Redundant Rules (Subscribed=No)**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rule 1 | Rule 2 | Confidence (Rule 1 / Rule 2) | Support (Rule 1 / Rule 2) | Redundant Reason |
| past\_marketing=unknown ⇒ subscribed=no | default\_credit=no, past\_marketing=unknown ⇒ subscribed=no | 0.908 / 0.908 | 33573 / 32862 | Rule 2 adds *default\_credit=no* but has the exact same confidence. |
| loan=no, past\_marketing=unknown ⇒ subscribed=no | default\_credit=no, loan=no, past\_marketing=unknown ⇒ subscribed=no | 0.902 / 0.902 | 27814 / 27371 | Rule 2 adds *default\_credit=no* but has the exact same confidence. |

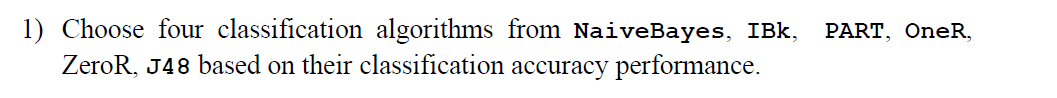
Although the above two pairs of rules got the same confidences, due to different supports, we do not classify them as redundant rules. However, they are strictly redundant so may provide little additional information when we are doing analysis.

The definition of redundant rules is that they provide little or no additional information. They do not affect the outcome or confidence levels, as they differ only slightly from other rules.

**Task 2**

**Weka Part:**

**Question1**



Answer:

A white sheet with black text

Description automatically generated

NaïveBayes accuracy is 94.2324%, IBK accuracy is 93.6687%, PART accuracy is 95.6418%, OneR accuracy is 95.2949%, ZeroR accuracy is 70.2732%, J48 accuracy is 95.8586%.

Based on accuracy, I have selected the following four classification algorithms: J48, PART, OneR, and NaiveBayes.

Weka process:

Step 1: open COVID19.arff in Preprocess

A screenshot of a computer

Description automatically generated

Step2: Choose different in classifier with cross validation Folds 10

A screenshot of a computer

Description automatically generated

Step3: Record the result.

**Question 2:**

A close up of a text

Description automatically generated A close up of text

Description automatically generated

Answer:

The best attributes for each classification algorithm are the minimum number of attributes that yield the highest accuracy. Here’s the optimal attribute selection for each algorithm:

For J48, the best attributes is 9, the accuracy is 95.6635%.

For PART, the best attributes is 9, the accuracy is 95.6418%.

For OneR, the best attributes is 3, the accuracy is 95.2949%.

For NaiveBayes, the best attributes is 3, the accuracy is 94.8395%

Here is the evidence:

A blue rectangular sign with black text

Description automatically generated A green rectangular box with black text

Description automatically generated

A yellow paper with black text

Description automatically generated A questionnaire with text on it

Description automatically generated with medium confidence

Among the four algorithms, J48 and PART provide the best performance, with only a slight difference between them. J48 achieves an accuracy of 95.66%, which is sligh higher than PART's accuracy of 95.64%, resulting in a difference of just 0.0217%.

Weka process:

Step 1: Go Classify page, select meta->AttributeSelectedClassifier from Classifier and use Cross-validation Folds 10

A screenshot of a computer

Description automatically generated

Step 2: Left click in AttributeSelectedClassifier

A close-up of a button

Description automatically generated

Step 3: Choose J48, PART, OneR, NaiveBayes in classifier and select InfoGainAttributeEval in evaluator and select Ranker in search.

A screenshot of a computer

Description automatically generated

Step4: Left click Ranker and try 3 to 10 in numToSelect

A screenshot of a computer

Description automatically generated

Step 5: Repeat all process until all result had been record(different classifier with numToSelect)

A screenshot of a computer

Description automatically generated

**Question3:**

A text on a white background

Description automatically generated

Answer:

Covid19 data set data is imbalanced, in attribute infection\_risk, the weight of high is 1371and the weight of low is 3241.

My cost matrix false negative rate had increase to 5, since I want to increase the impact if the infection risk had been predicted “low” but actually it is “high”.

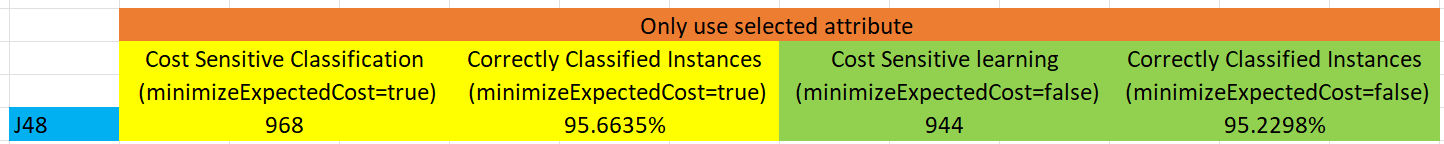
A screenshot of a computer

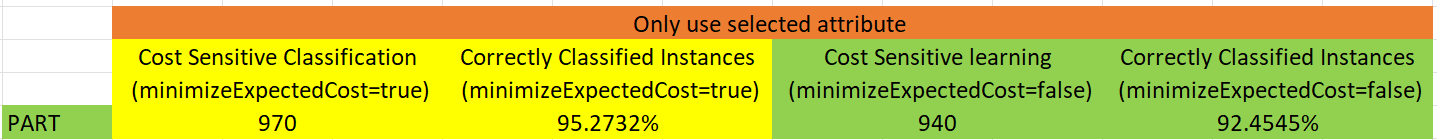
Description automatically generated

J48 and PART are both decision tree principles, and there are producing similar tree structures in the same dataset.

The algorithm I would like to use in order to minimize the cost is J48.

J48:



PART:

If the minimizeExpectedCost=false, J48 Total cost is 944 and accuracy is 95.2298%, PART Total cost is 940 and accuracy is 92.4545%, There of J48 is better in accuracy but worst in total cost.

When the minimizeExpectedCost=true, J48 have a total cost 968 with accuracy 95.6635%, PART total cost is 970 with accuracy 95.2732%. There of J48 is better.

Final conclusion: I believe J48 is the better option because, when minimizing the classification errors for the "infection\_risk = high" class, it has a lower total cost and a higher number of correctly classified instances with better accuracy.

Weka Process:

Step 1: In Classifiy page, more options button, set Cost Matrix to 2 classes and increase the false negative to 5.

A screenshot of a computer

Description automatically generated

Step 2: Choose Classifier, meta-> AttributeSelectedClassifier.

A screenshot of a computer

Description automatically generated

Step 3: Use InfoGainAttributeEval, Ranker with selected number in question 2 in numToSelect (I have selected “9”), and choose classifier CostSensitiveClassifierA screenshot of a computer

Description automatically generated

Step 4: Choose different classifier(J48, PART) and costMatrix set classes =2, and false negative to 5. Use explicit cost matrix in costMatrixSource

A screenshot of a computer

Description automatically generated

Step 5: Try minimizeExpectedCost = true or false.





Step 6: Record the outcome.

A screenshot of a computer

Description automatically generated

**Task 2**

**Java Part:**

**Question 1:**

**A text on a white background

Description automatically generated**

Answer:

All my Java code outcome is the same with my question 2 outcome.

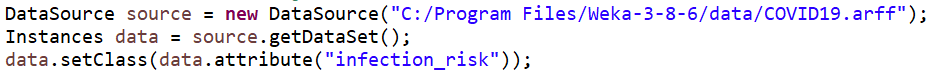
Here is a summary of my Java output. I then provide evidence based on each classifier to support the outcomes.

A screenshot of a computer

Description automatically generated

For all the algorithms, I used the same COVID-19 dataset and set the target class to infection\_risk. I then used the AttributeSelectedClassifier with classifiers (J48, PART, OneR, NaiveBayes), and evaluated attributes with InfoGainAttributeEval Ranker. I adjusted the ranker to either 9 or 3, and finally, printed the performance results for each classifier.

Here is some code for your reference.













A computer code with text

Description automatically generated

So based on the performance J48 and PART is the best.

Evidence:

J48 classifier:

A close up of a word

Description automatically generated

For J48, I have selected 9 attributes, and the output is the same with my question 2, my accuracy is 95.6635, and the correctly classified instances value is 4412

A number and percentage symbol

Description automatically generated with medium confidenceA blue rectangular sign with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

PART classifier:

A close up of a word

Description automatically generated

For PART, I also selected 9 attributes, achieving an accuracy of 95.6418% with 4411 correctly classified instances.

A number with black text

Description automatically generated with medium confidenceA green rectangular box with black text

Description automatically generatedA screenshot of a computer

Description automatically generated

OneR classifier:

A close up of a word

Description automatically generated

With the OneR algorithm, only 3 attributes were selected. This yielded an accuracy of 95.2949% and 4,395 correctly classified instances. A number on a white background

Description automatically generatedA yellow paper with black text

Description automatically generatedA screenshot of a computer

Description automatically generated

NaiveBayes classifier:

A close up of a text

Description automatically generated

We have selected 3 attributes to NaiveBayes, It accuracy is 94.8395 and correctly classified instances value is 4374.

A number on a white background

Description automatically generatedA questionnaire with text on it

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated

**Java Part:**

**Question 2:**

A close up of a text

Description automatically generated

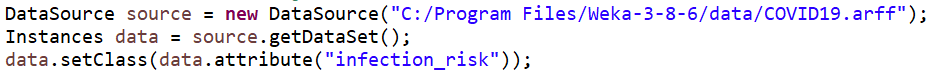
In question 1.3 I have use AttributeSelectedClassifier with CostSensitiveClassifier together and select InfoGainAttirbuteEval and Ranker 9. And then I selected J48 and PART as my classifier in CostSensitiveClassifier. And my CostMatrix false negative is 5. The Java summary is showing that not matter MinimizeExpectedCost is false or true, my classifier accuracy and total cost is the same with Weka part question 3.

A screenshot of a computer

Description automatically generated

Answer:

Step 1: Use original data COVID19.arff



Step 2: Use AttributeSelectedClassifier with CostSensitiveClassifier together and select InfoGainAttirbuteEval and Ranker 9

A screenshot of a computer program

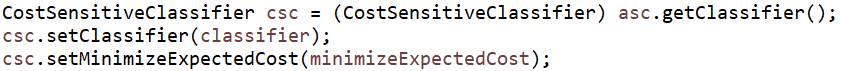
Description automatically generated

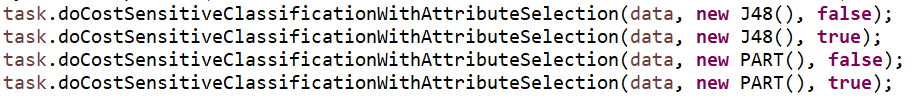
Step 3: Set false negative rate to 5 in costMatrix.

A close up of a number

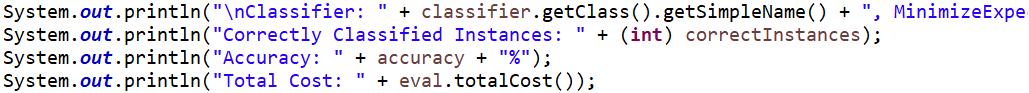
Description automatically generated

Step 4: CostSensitiveClassifier with different classifier and use true or false in minimizeExpectedCost





Step 5: Record the outcome.



Evidence:

J48 classifier:



Of minimizeExpectedCost is false, The Accuracy is 95.2298%, and the Total cost is 944.

A black text on a white background

Description automatically generated

A green and orange rectangular sign with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

Of minimizeExpectedCost is true, The Accuracy is 95.6635%, and the Total cost is 968.

A black text on a white background

Description automatically generated

A yellow and black sign with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

PART classifier:



A total cost is 940 with a correctly classified instances accuracy 92.4545% when minimizeExpectedCost = false

A black text on a white background

Description automatically generated

A green and orange rectangular box with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

A total cost is 970 with a correctly classified instances accuracy 95.2732% when minimizeExpectedCost = true

A black text on a white background

Description automatically generated

A yellow and black sign

Description automatically generated

A screenshot of a computer

Description automatically generated

## Task 3: Text classification in Weka and Java (10 marks)

Download dataset **News.arff**. This is a text dataset consisting of 14,018 news documents. These news documents are categorised into four classes: computer, politics, science, and sports. In this task, you are required first to classify the news documents using Weka to determine some parameters in the filter, then develop a Java program to build a classifier to classify the news in this dataset.

### 1. Attribute selection in Weka

In this part, you need to use a filter in Weka to select attributes from the documents. You can choose 100 attributes and use J48 classifier to do the classification. For the parameters in the filter, you can use their default values or you can set up some values of your choice. You are required to tune 4 or 5 parameters that you think are important for determining the attributes.

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

**Answer:**

Preprocess filter “StringToWordVector” is chosen.

A screenshot of a computer

Description automatically generatedIf we don’t change any default parameters setting and just change wordToKeep to 100 (attributes), the accuracy is around 58.696%

Below is the output for the parameter changes:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parameter setting** | **Classifier** | **Test Option** | **Correctly Classified Instances** | **Incorrectly Classified Instances** | **Accuracy (%)** | **Incorrectly Classified (%)** |
| Default value | J48 | Cross-validation; 10 folds | 8228 | 5790 | 58.696 | 41.304 |
| Added Lovins Stemmer | 9370 | 4648 | 66.843 | 33.157 |
| Added Lovins Stemmer + Rainbow stopwords | 10754 | 3264 | 76.716 | 23.284 |
| Added Lovins Stemmer + Rainbow stopwords, change outputWordCounts to 'true' | 10770 | 3248 | 76.83 | 23.17 |
| Added Lovins Stemmer + Rainbow stopwords, change outputWordCounts to 'true', TFTransform change to 'True' | 10779 | 3239 | 76.894 | 23.106 |
| Added Lovins Stemmer + Rainbow stopwords, change outputWordCounts to 'true', TFTransform change to 'True', IDFTransform change to 'True' | 10787 | 3231 | 76.951 | 23.049 |
| Added Lovins Stemmer + Rainbow stopwords, change outputWordCounts to 'true', TFTransform change to 'True', IDFTransform change to 'True', 6. normalizeDocLength to Normalize all data | 10630 | 3388 | 75.8311 | 24.1689 |

Below are the reasons we choose to change the parameter values.

1. **Stemmer**:

Chosen Value: **LovinsStemmer**

Reason: Applying stemming reduces words to their root form, improving generalization by grouping similar words.

1. **StopwordsHandler**:

Chosen Value: **Rainbow**

Reason: Stopwords (common, unimportant words) were removed using the Rainbow stopwords list, which helps focus on the more meaningful words.

1. **outputWordCounts**:

Chosen Value: **True**

Reason: Enabling this option ensures that word frequency (how often a word appears) is considered, which can help in text classification as frequent words in certain contexts may be more important.

1. **TFTransform**:

Chosen Value: **True**

Reason: This transforms the word counts to Term Frequency (TF), scaling the word counts relative to the total word count in the document. This helps normalize the document lengths.

1. **IDFTransform**:

Chosen Value: **True**

Reason: Inverse Document Frequency (IDF) reduces the impact of very common words that appear across many documents, highlighting more unique words.

1. **normalizeDocLength**

Chosen Value: **Normalise all data**

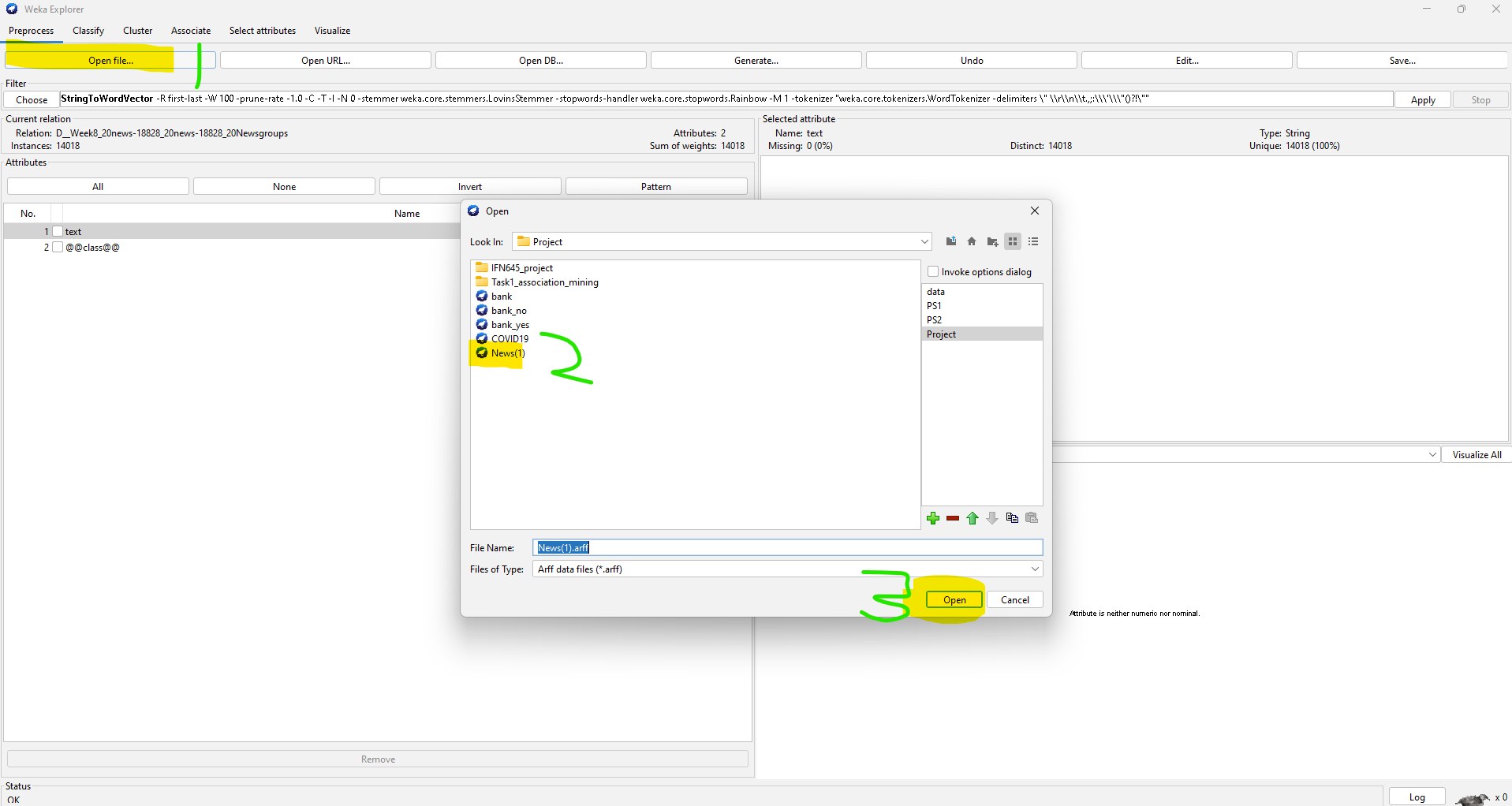
Reason: Ensures documents of varying lengths contribute equally by scaling word counts, preventing longer documents from dominating the feature space. This balances the importance of short and long documents in classification.

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

**Answer:**

Weka Explorer is used to preprocess the dataset and apply a filter. Below is a step-by-step outline of the process followed to determine the values for the parameters in the filter:

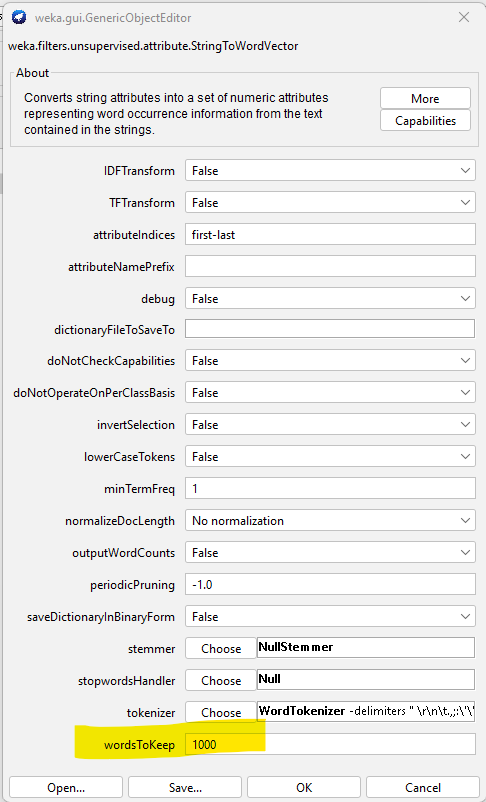
1. **Step 1**: Press the **"Open file"** button, select the file named **"news.arff"**, then press **"Open"** to load the dataset into Weka.



A screenshot of a computer

Description automatically generated**Step 2**: Press the **"Choose"** button under the **Filter** section, and select the **StringToWordVector** filter from the list of available filters.

**Step 3**: In the **Weka ObjectEditor**, set the **wordsToKeep** parameter to **100** (reduced from the default value of 1000) to select 100 attributes. Other settings are kept as default. After making the changes, press **"OK"**, and then press **"Apply"** to apply the filter to the dataset.



**Step 4**: Go to the **"Classify"** tab, select the **J48** classifier, and keep the default settings (cross-validation with 10 folds). Press **"Start"** to begin the classification process.

A screenshot of a computer

Description automatically generated

A white background with black text

Description automatically generated **Step 5**: After the classification task completes, the results are displayed in the **Classifier Output** section.

A screenshot of a computer

Description automatically generated Next, repeated the classification process (starting from Step 1), changing one parameter each time to evaluate its effect on the accuracy of the classification task. For example, parameters such as the **stemmer**, **stopwordsHandler**, **outputWordCounts**, **TFTransform**, **IDFTransform** and **normalizeDocLength** were progressively modified to observe their impact on performance.

### 2. Java program

For this part, you are required to develop a Java program to classify the documents in the news dataset.

1. Perform the classification task using 4 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.
2. Your program should display the correctly classified instances results, accuracy, and the time taken by each algorithm.
3. Which classifier performs the best in terms of time efficiency? Describe why this algorithm is faster than others.

**Answer:**

**Java output:**

Classifier: IBk

Correctly Classified Instances 11547 82.3727 %

Incorrectly Classified Instances 2471 17.6273 %

Kappa statistic 0.7573

Mean absolute error 0.0898

Root mean squared error 0.2816

Relative absolute error 24.5608 %

Root relative squared error 65.8823 %

Total Number of Instances 14018

Correctly classified instances: 11547.0

Incorrectly classified instances: 2471.0

Time taken: 94289ms

===================================================

Classifier: SMO

Correctly Classified Instances 11826 84.363 %

Incorrectly Classified Instances 2192 15.637 %

Kappa statistic 0.7852

Mean absolute error 0.2673

Root mean squared error 0.338

Relative absolute error 73.1302 %

Root relative squared error 79.0642 %

Total Number of Instances 14018

Correctly classified instances: 11826.0

Incorrectly classified instances: 2192.0

Time taken: 165954ms

===================================================

Classifier: J48

Correctly Classified Instances 10873 77.5646 %

Incorrectly Classified Instances 3145 22.4354 %

Kappa statistic 0.6928

Mean absolute error 0.1221

Root mean squared error 0.3159

Relative absolute error 33.398 %

Root relative squared error 73.8928 %

Total Number of Instances 14018

Correctly classified instances: 10873.0

Incorrectly classified instances: 3145.0

Time taken: 382459ms

===================================================

Classifier: HoeffdingTree

Correctly Classified Instances 10984 78.3564 %

Incorrectly Classified Instances 3034 21.6436 %

Kappa statistic 0.7021

Mean absolute error 0.1087

Root mean squared error 0.326

Relative absolute error 29.7482 %

Root relative squared error 76.2602 %

Total Number of Instances 14018

Correctly classified instances: 10984.0

Incorrectly classified instances: 3034.0

Time taken: 45589ms

===================================================

**Comparison of 4 algorithms:**

|  |  |  |  |
| --- | --- | --- | --- |
| Classifier | Correctly Classified Instances | Accuracy (%) | Time Taken (ms) |
| IBk | 11547 | 82.37 | 94289 |
| SMO | 11826 | 84.36 | 165954 |
| J48 | 10873 | 77.56 | 382459 |
| HoeffdingTree | 10984 | 78.36 | 45589 |

From the table above, we note that HoeffdingTree performs the best in terms of time efficiency, being significantly quicker than the other classifiers.

The reason for this time efficiency lies in its algorithms, which are designed for streaming data and incremental learning. HoeffdingTree can work with a small, fixed amount of data to build trees, so it does not need to process all instances multiple times. It also uses approximate statistics to make decisions, which reduces computational complexity and leads to faster training times.

A black and white math equation

Description automatically generatedThe number of samples required to guarantee, with high confidence, that the characteristic selected for a split based on the sample would be the same as if the complete dataset were utilized is known as the Hoeffding bound. Before reaching a split

A close-up of a white background

Description automatically generated

decision, it makes sure that there is a statistically significant difference between the best and second-best traits.

On the other hand, the other three algorithms (IBk, SMO, and J48) require full tree building or involve solving quadratic optimization problems, which demand more computational resources and result in longer processing times.

HoeffdingTree’s speed and efficiency in handling data streams make it the fastest algorithm in this task. Its ability to incrementally update and approximate splits quickly allows it to outperform more resource-heavy algorithms like J48, IBk, and SMO.