Task A

Question 1:
The two frequent pattern mining algorithms used where FP Growth and Apriori. For all three minimum supports (0.2, 0.3, 0.4), FP Growth preformed more efficiently regarding time at all three support levels compared to Apriori. The results can be seen in *table 1*. FP Growth uses a tree structure for the data, appose to Apriori's database structure which makes it a much faster approach, as Apriori must do multiple scans of the database to find frequent rules.

Table 1: Algorithm Run Time

Minimum Support Level	FP Growth Total Runtime (Milliseconds)	Apriori Total Runtime (Milliseconds)
0.2	106	394
0.3	65	166
0.4	72	82

Question 2:
Frequent pattern mining was preformed using FP Growth as it was the most efficient in the previous testing, support was set to 0.2 (20%) for testing on both datasets. Across both the Yes-class and No-class datasets there where many attributes that featured in both datasets top 5 frequent patterns. These attributes include default_credit=no, loan=no, call_duration=100-500s, past_marketing=unknown, marital=married. The only attribute unique to any dataset was age=20-30s, which only appeared in the Yes-class dataset. Default_credit=no and loan=no were the two most common occurring attributes in the top 5 frequent patterns for the Yes-class, with a confidence of 1 (100%) when default_credit=no is considered the rule head and loan=no is considered the rule body. Similarly, is seen in No-class were default_credit=no and loan=no is tied with the highest confidence along with default_credit=no and past_marketing=unknown, with a confidence of 0.6 (60%).

Table 2: Size-3 Most Frequent Patterns

Dataset	Attribute 1	Attribute 2	Attribute 3	Support
Yes-class	default_credit=no	housing=no	loan=no	3120
Yes-class	default_credit=no	loan=no	past_marketing=unknown	2988
Yes-class	default_credit=no	loan=no	call_duration=100-500s	2712
Yes-class	age=21-30s	default_credit=no	loan=no	2519
Yes-class	marital=married	default_credit=no	loan=no	2471
Dataset	Attribute 1	Attribute 2	Attribute 3	Support
No-Class	default_credit=no	loan=no	past_marketing=unknown	27371
No-Class	default_credit=no	call_duration=100-500s	past_marketing=unknown	21352

No-Class	default_credit=no	loan=no	call_duration=100-500s	21217
No-Class	default_credit=no	past_marketing=unknown	marital=married	20357
No-Class	default_credit=no	loan=no	marital=married	19799

NEXT PAGE FOR FURTHER QUESTION

Question 3:
Using the doMaxFP algorithm with support set to 0.2 (20%) the largest frequent patterns were generated from both the Yes-class and No-Class separately. The patterns generated were outputted in a text file and analyzed using python *(the python script is included in the submission folder)* to establish largest patterns and order in support. For both datasets the largest pattern featured 5 attributes, although the Yes-Class dataset only produced 2 patterns with 5 attributes with the other three only featuring 4. However, it should be noted that the Yes-class dataset was much smaller (5,289 patterns) compared to No-class (39,922 patterns), this could possibly give reason for No-class have more patterns of a larger value above the support threshold of 0.2. default_credit=no and loan=no with 1 (100%) confidence in both datasets. When default_credit=no and loan=no is looked at as the head rule and past_marketing=unknown is looked at as the body rule confidence is still 1 (100%) in the No-class data, while confidence is only 0.6 (60%) in the Yes-class dataset. Housing and balance where the only unique attributes among the datasets, with balance only featuring in the No-class dataset with balance=bellow-1k featuring in the third most frequent pattern. Housing=yes also only featured in the No-class dataset, where housing=no only featured in the Yes-class dataset.

Table 3: Top 5 Largest Frequent Patterns

Dataset	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Support
No-class	default_credit=no	past_marketing=unknown	loan=no	call_duration=100-500s	marital=married	10813
No-class	default_credit=no	past_marketing=unknown	loan=no	call_duration=100-500s	housing=yes	10020
No-class	default_credit=no	past_marketing=unknown	loan=no	call_duration=100-500s	balance=below-1k	9284
No-class	default_credit=no	past_marketing=unknown	loan=no	marital=married	housing=yes	9236
No-class	default_credit=no	past_marketing=unknown	loan=no	call_duration=100-500s	age=21-30s	9131
Dataset	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Support
Yes-Class	default_credit=no	loan=no	past_marketing=unknown	housing=no	call_duration=100-500s	1124
Yes-Class	default_credit=no	loan=no	housing=no	call_duration=100-500s	marital=married	1075

Yes-Class	default_credit=no	loan=no	past_marketing=unknown	age=21-30s	NULL	1586
Yes-Class	default_credit=no	loan=no	housing=no	age=21-30s	NULL	1506
Yes-Class	default_credit=no	loan=no	past_marketing=unknown	marital=married	NULL	1502

NEXT PAGE FOR FURTHER QUESTION

Question 4

Apriori is the slowest across all minimum supports, which is not to surprising, while it is a historically significant algorithm it is not the most modern, nor does it employ modern techniques. Charm Frequent Closed was the fastest at all supports, this is likely down to its implementation of depth-first search for mining frequent closed itemset and is one of the most efficient algorithms in the SPMF library (Fournier-Viger, 2022). FC Growth also remains more efficient then Apriori due to its triangular matrix used appose to Apriori's use of a relational database structure that needs multiple scans, making it slower to develop pairs then the other two methods.

Table 4: Closed Algorithms at Min Support Levels

Algorithm	Minimum Support 0.2	Minimum Support 0.3	Minimum Support 0.4
Apriori Frequent Closed	335 ms	138 ms	62 ms
FC Growth Frequent Closed	73 ms	51 ms	42 ms
Charm Frequent Closed	45 ms	22 ms	17 ms

Question 5

The top 10 frequent association rules with subscribed=yes and subscribed=no as the consequent were mined using the AlgoTopKClassRules algorithm from the SPMF package. The algorithmic takes the consequent in integer value, so 11 was passed for subscribed=no and 42 was passed for subscribed=yes, as that's how they were encoded. The bank.arff was used as the data set and the results can be seen in *Table5*. *Table6* shows the non-redundant rules returned from the AlgoTopKClassRules. Rules were determined redundant if they could be assumed by other rules Prescence in the top 10 list. In the case of Subscribed=no as the consequent, Default_credit=no, call_duration=100-500s => subscribed=no, this could already be determined because of the presence default_credit=no => subscribed and call_duration=100-500s => subscribed=no. The criterion for redundant rules is all based on the assumption that the minimum support of 0.2 is the only criteria. If a confidence rule was included the non-redundant rules may change, for example for subscribed=yes as a consequent call_duration=500-1k => subscribed=yes is a rule at *0.38* confidence but if the confidence threshold was raised to 0.39 it would no longer be a rule and call_duration=500-1k, loan=no => subscribed=yes would now be a non-redundant rule. In the case of the second chart where Subscribed=no is the consequent there were no rules that could be deemed as redundant in a similar manor to when consequent was Subscribed=yes, however there are potential rules that could be deemed redundant based on your perception of rule redundancy. For example, Marketing_success => subscribed=yes is a rule with 978 support and 0.647 confidence, but then the rule Marketing_success, Default_credit=no => Subscribed= yes has the exact same support but only a marginally better confidence at 0.6481. With a rule like this you could argue that adding Default_credit=no to the rule adds very little information to the preexisting rule and in cases where complexity wants to be reduced for computational efficiency you could deem this rule redundant. Another example is the default_credit=no, loan=no, past_marketing=unknown, call_duration=500-1k => subscribed=yes having a lower support and confidence then the base rule of call_duration=500-1k => Subscribed=yes, suggesting that the other additions to the rule are not improving the strength of the original rule. Of course, the more attributes in a rule the more likely it is that support and confidence will decrease but again these rules can be potentially be deemed redundant if rules are trying to be kept to a minimum for efficiency reasons, but generally they can be included and add insight that the single rules may not be able to provide. If we assume the yellow highlighted colors to be non-redundant then there are more truly redundant rules when subscribe-yes is the consequent. For example, Default_credit=no, loan=no, past_marketing=unknown, call_duration=500-1K -> subscribed=yes is redundant as loan=no, past_marketing=unknown, call_duration=500-1k -> subscribed=no and default_credit=no, past_marketing=unknown, call_duration=500-1k -> subscribed=no exist in the list and make up all the components of the redundant rule. To clarify in the case were the existing rules are call_duration=500-1k, loan= no => subscribed=yes and call_duration=500-1k=>subscribed=yes, loan=no => subscribed=yes will not be considered a rule because there is no evidence that, that rule is efficient in any metric on its own, but depending on interpretation it could be considered a rule making call_duration=500-1k, loan= no => subscribed=yes redundant with the presence of loan= no => subscribed=yes and call_duration=500-1k=>subscribed=yes.

Table5: Top 10 Association Rules

Subscribed=no as Consequent						Support	Confidence
Marital=married	=>	Subscribed=no				24,459	0.899
Default_credit=no	Call_duration=100-500s	=>	Subscribed=no			25,538	0.899
Call_duration=100-500s	=>	Subscribed=no				26,044	0.9
Default_credit=no	Loan=no	Past_marketing=unknown	=>	Subscribed=no		27,371	0.902
Loan=no	Past_marketing=unknown	=>	Subscribed=no			27,814	0.902
Default_credit=no	Loan=no	=>	Subscribed=no			32,685	0.873

Default_credit=no	Past_marketing=unknown	=>	Subscribed=no			32,862	0.908
Loan=no	=>	Subscribed=no				33,162	0.873
Past_marketing=unknown	=>	Subscribed=no				33,573	0.908
Default_credit=no	=>	Subscribed=no				39,159	0.882
Subscribed=yes as Consequent						Support	Confidence
past_marketing=success	=>	subscribed=yes				978	0.647
default_credit=no	past_marketing=success	=>	subscribed=yes				
default_credit=no	loan=no	past_marketing=unknown	call_duration=500-1k	=>	subscribed=yes	1,061	0.359
loan=no	past_marketing=unknown	call_duration=500-1k	=>	subscribed=yes		1,079	0.36
default_credit=no	past_marketing=unknown	call_duration=500-1k	=>	subscribed=yes		1,214	0.349
past_marketing=unknown	call_duration=500-1k	=>	subscribed=yes			1,242	0.35
default_credit=no	loan=no	call_duration=500-1k	=>	subscribed=yes		1,428	0.393
loan=no	call_duration=500-1k	=>	subscribed=yes			1,448	0.394
default_credit=no	call_duration=500-1k	=>	Subscribed=Yes			1,614	0.379
call_duration=500-1k	=>	subscribed=yes				1,646	0.38

Table 6: Redundant Rules are Highlighted

Subscribed=no as Consequent						Support	Confidence
Marital=married	=>	Subscribed=no				24,459	0.899
Default_credit=no	Call_duration=100-500s	□ =>	Subscribed=no			25,538	0.899
Call_duration=100-500s	=>	Subscribed=no				26,044	0.9
Default_credit=no	Loan=no	Past_marketing=unknown	□	Subscribed=no		27,371	0.902
Loan=no	Past_marketing=unknown	=>	Subscribed=no			27,814	0.902
Default_credit=no	Loan=no	=>	Subscribed=no			32,685	0.873
Default_credit=no	Past_marketing=unknown	=>	Subscribed=no			32,862	0.908
Loan=no	=>	Subscribed=no				33,162	0.873
Past_marketing=unknown	=>	Subscribed=no				33,573	0.908
Default_credit=no	=>	Subscribed=no				39,159	0.882
Subscribed=yes as Consequent						**Support**	**Confidence**
past_marketing=success	=>	subscribed=yes				978	0.647
default_credit=no	past_marketing=success	=>	subscribed=yes			978	0.6481

default_credit=no	loan=no	past_marketing=unknown	call_duration=500-1k	=>	subscribed=yes	1,061	0.359
loan=no	past_marketing=unknown	call_duration=500-1k	=>	subscribed=yes		1,079	0.36
default_credit=no	past_marketing=unknown	call_duration=500-1k	=>	subscribed=yes		1,214	0.349
past_marketing=unknown	call_duration=500-1k	=>	subscribed=yes			1,242	0.35
default_credit=no	loan=no	call_duration=500-1k	□	subscribed=yes		1,428	0.393
loan=no	call_duration=500-1k	=>	subscribed=yes			1,448	0.394
default_credit=no	call_duration=500-1k	=>	Subscribed=Yes			1,614	0.379
call_duration=500-1k	=>	subscribed=yes				1,646	0.38

Task 2

Classification in Weka

Question 01

We uploaded the covid19_risk.arff file in the weka and the next setups of the process were like the followings –

Classifier: AttributeSelectedClassifier
Evaluator: InfoGainAttributeEval (Single Attirbute Filter)
Search: Ranker
Cross-validation: 10 folds

After running the classification algorithms mentioned in the question, we got the following results before and after having some tuning –

Algorithm	**% Correctly Classified Instances**
Naïve Bayes	87.3802
Logistic	89.1411
IBk	85.5997
PART	88.0063
ZeroR	63.4122
J48	89.3954

From this table, we can see that Naïve Bayes, Logistic, PART and J48 performed the best among all.

Question 02

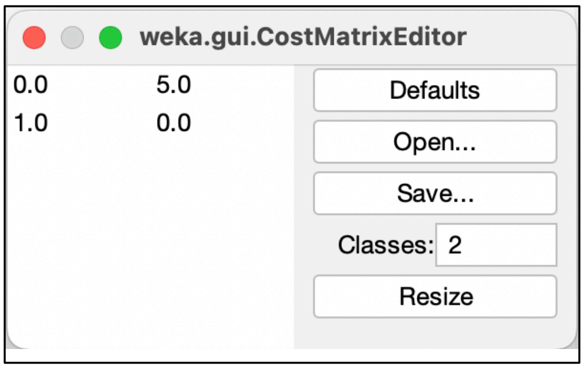
To achieve a relatively better classification performance for each algorithm, we changed the number of attributes to be used with the option "numToSelect" in Ranker and used 3 to 10 attributes for testing.

Algorithm	**% Correctly Classified Instances (3-10 attributes)**	**% Correctly Classified Instances (10< attributes)**
Naïve Bayes	88.7498 (3 attributes)	Not needed
Logistic	89.0432 (6 attributes)	89.1606 (17 attributes)
PART	88.9845 (6 attributes)	Not needed
J48	89.0041 (3 attributes)	89.4541 (14 attributes)

We observed that for J48 the performance decreases from the default settings and the highest accuracy it achieved was 89.0041% for all the numbers of attributes from 3 to 7. However, for 14 attributes it reached the highest. Same goes for Logistic.
In short, we can say that in the mentioned range of number of attributes 3-10, the best performer was Logistic with 89.0432% accuracy with 6 attributes but J48 performed best when the number of attributes was chosen beyond the range. Its accuracy was 89.4541% with 14 attributes.
On the other hand, Naïve Bayes performed better than default with 3 attributes and 88.7498% accuracy and the accuracy of PART was 88.9845% with 6 attributes.

Question 03

Based on the performances mentioned in the Question 2, we chose J48 and Logistic for cost sensitive analysis.
For the cost matrix, we have put 5 for the false negative values since infection_risk=high was more important. The cost matrix is the following –



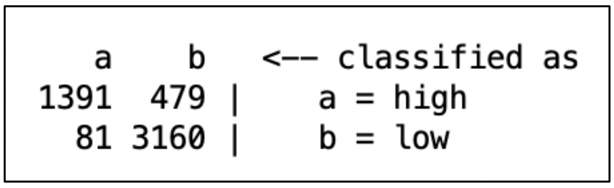
0.0	5.0
1.0	0.0

After the modification, the cost evaluation was like the following table –

Classifier	Base	Cost Sensitive	Cost Sensitive Learning (minimizeExpectedCost=true)	Cost Sensitive Classification (minimizeExpectedCost=false)
Logistic	560	2476	1969	1951
J48	562	2610	2437	1954

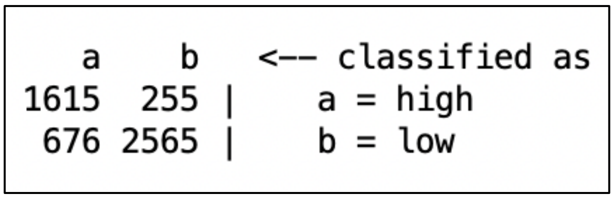
All instances of Cost Sensitive Learning and Cost Sensitive Classification are lower than that of Cost Sensitive analysis. Cost Sensitive Learning preforms better then Cost Sensitive Classification for both algorithms, making it the more suitable choice between the two methods.

Now if we look into the performance, we can see that even though the false negative were reduced, the accuracy dropped for Logistic (from 89.0432% to 81.7844%) –



```
    a    b   <-- classified as
 1391  479 |    a = high
   81 3160 |    b = low
```

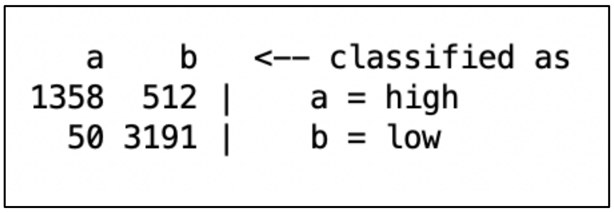
Figure: Before the cost sensitive classification for logistic.



```
    a    b   <-- classified as
 1615  255 |    a = high
  676 2565 |    b = low
```

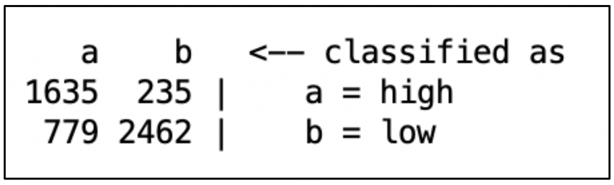
Figure: After cost sensitive classification for logistic

For J48, the scenario was same. The false negatives were reduced as well as the accuracy (from 89.0041% to 80.1604%) –



```
   a    b   <-- classified as
1358  512 |    a = high
  50 3191 |    b = low
```

Figure: Before cost sensitive classification for J48



```
   a    b   <-- classified as
1635  235 |    a = high
 779 2462 |    b = low
```

Figure: After the cost sensitive classification for J48

Therefore, considering the performance regarding accuracy, we would like to use Logistic to minimize the cost since it performed better than J48.

Task 3
Question 1
J48 was set to standard parameters with 10-fold cross-validation.

The following are results from the base string to word vector filter and parameter tuned versions.

Original results
Base String to word vector: No Parameter tuning
Correctly Classified Instances 2255 80.4208 %
Incorrectly Classified Instances 549 19.5792 %
Time taken to build model: 728.53 seconds

Results of Chosen Filter
String to word vector: Love Stemmer, stop words Rainbow, WordTokenizer, words to keep=100, NormalizeDocLenght=Normalize all data, minTermFreq = 1
Correctly Classified Instances 2104 75.0357 %
Incorrectly Classified Instances 700 24.9643 %
Time taken to build model: 16.27 seconds

Without parameter tunning J48 produced its best accuracy but took over 10 minutes to run. The next step was to tune the parameters. The first iteration of tuning was to use a stemmer, stop words, word tokenizer, set words to keep to 100 and normalize all the data by document length. The use of the word stemmer was chosen because it will reduce the chance of keeping variations of words as different attributes (i.e., eating, eaten, eats is just eat). Rainbow stop words seemed to work the best when compared to multi stop words. It is likely that multiword stop word had included stop words that may be related to the article class content as the model preformed worse when this was used. Word tokenizer was tuned as it controls the way words and sentences are broken up into attributes. I thought N-grams would work best as it is commonly used in machine learning, but it seemed to make for a worse result. Words to keep was set to 100 as per the question requirements. Normalization of document length was set to all data. This is a method that can be seen in many text-based algorithms such as TF-IDF. There was very minimal improvement in the classifier, as most documents are likely the same length. The final parameter that was tuned was minimum term frequency. We found that leaving this in its standard parameter of 1 produced the best results, as increasing this parameter produced diminishing returns. The final result is what is shown above. It does not produce as high accuracy but it's time to run is substantially lower. However more improvements could be made as the stop words do not include special characters and many feature as attributes that likely don't produce much information unless they are specific to computer class articles and are from programing languages or similar context.

Question 2
Out of the four algorithms the Hoeffding Tree preformed the best with a runtime of 57,313 milliseconds. It should be noted that all runtimes include the filtering runtime preformed on the data, but as all parameters were kept the same for all the filters used on each algorithm so there shouldn't be any bias between the algorithms based on filter. The second fastest algorithm was the IBK classifier, followed by the SMO Classifier and then the J48 Classifier that was the slowest by a large margin.

The Hoeffding Tree was the best preforming in terms of time efficiency, this is likely due to its use of Hoeffding bound, which allows for the algorithm to choose an optimal splitting attribute from a small sample and produce similar results to the typical decision tree method. It is not specified if the Weka implementation of Hoeffding tree is built off a subset of the data (I could not find any mention to this in the documentation) but it would make sense if it was only using a subset of the dataset as it built a decision tree much faster than the other algorithms. Like the Hoeffding tree algorithm the IBK algorithm is an incremental classification algorithm. This means that the algorithms use batch settings and build the classifier with the dataset in the memory. Using the memory is much faster than having to store the entire dataset in storage, which would have to be done in the case of J48 deciding tree, this is likely the reason J48 took so long, also the number of attributes do not lead their self to traditional decision trees as it takes a long time to structure the whole tree.

Algorithm	Correctly Classified Instances	Incorrectly Classified Instances	Accuracy	Time taken to Run (ms)
IBK Classifier	10,823	3,195	77.21%	111,240
SMO Classifier	11,489	2,529	81.96%	303,323
J48 Classifier	10,581	3,437	75.49%	326,516
Hoeffding Tree Classifier	10,812	3,206	77.13%	50,084