IFN645 Large Scale Data Mining

Semester 2, 2024

Capstone Assignment

Due date: Monday, 21 October 2024, 11:59pm
Weighting: 35%
Team: Kate Klibbe - Nima Ghamari

Required to be submitted on Canvas:
One single zip file which contains the following files/folders:

1. A report (pdf or Word file) containing your answers to the questions specified belowand a statement of completeness stating which tasks have been completed.
2. Three separate folders each containing one Java project for one of the three tasks.

Statement of Completeness

Question	Status
Task 1:	
1. Dataset	
2. Questions	
1.	Complete
2.	Complete
3.	Complete
4.	Complete
Task 2:	
1. Classification in Weka	
1.	Complete
2.	Complete
3.	Complete
2. Java program	
1.	Complete
2.	Complete
Task 3:	
1. Attribute selection in Weka	
1.	Complete
2.	Complete
2. Java program	
1.	Complete
2.	Complete
3.	Complete

Introduction

This assignment is intended to allow you to display your knowledge and understanding developed in lectures and practicals. The purpose of this assignment is to give you (1) an understanding that various methods can be applied to different types of datasets such as text and transactions, and (2) the benefits of applying data analytics techniques to a data domain.

There are three tasks in this assignment. You need to complete all the tasks.

Task Specification

Task 1: Association mining in Java (12 Marks)

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.

1. **Dataset**

   Download the datasets **`bank.arff`**, **`bank_no.arff`**, and **`bank_yes.arff`** from the Canvas. **`bank.arff`** is the entire dataset collected in the campaign which consistsof 45,211 records. Each record in the dataset is about one customer described by 11 attributes. After this marketing campaign, a customer could subscribe the term-depositproduct or not, indicated by attribute 'subscribed'. This attribute divides customers intotwo classes, i.e., yes-class and no-class. **`bank_no.arff`**, and **`bank_yes.arff`** aretwo subsets of the entire dataset containing customers in each of the two classes, respectively.

   To help you understand the data, the following table provides you with the descriptionto each of the attributes.

Attribute	Description
age	Customer age
job	Customer job type
marital	Customer marital status
education	Customer education status
default_credit	Whether customer has credit in default or not
balance	Customer bank account balance
housing	Whether customer has housing/home loan or not
loan	Whether customer has personal loan or not
call_duration	Phone contact duration in seconds, e.g., 500-1k indicatesthat the phone talk with the customer lasts 500 to 1000 seconds.
past_marketing	Outcome of last marketing to this customer, e.g., failure, indicates that the last marketing was unsuccessful to this customer.
subscribed	Whether customer subscribes the term-deposit product or not.

In the context of this task, an item is an attribute with a specific value, e.g., **`age=40s`**,and a pattern is a set of items, e.g., {**`default_credit=no, loan=no, age=21-30s`**} is a size-3 pattern. The patterns generated from the bank dataset describe the characteristics or behaviour of the customers in that dataset.

After the marketing campaign, the bank wants to derive some information from the dataset. Specifically, they want to know the popular patterns in each customer class andalso want to know the associations between customer behaviour and each class, i.e., theassociation rules that represent the implications from customer attributes to one of the classes. As a data analyst, you are required to develop a Java program that can generatepatterns and association rules from the datasets.

2. 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 timeefficiency. You can use different minimum supports, e.g., 0.2, 0.3, and 0.4, to do the comparison. Show your comparison result.

2) Choose a frequent pattern mining algorithm based on the comparison in 1), state thereason to choose this algorithm. Then use the chosen algorithm to generate the top5 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 commoncharacteristics and differences if there are any.
   Specify the minimum support that you have used for this question.

3) 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 ifthere are any.
Specify the minimum support that you have used for this 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 minimumconfidence 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.

Q1:
At a 0.2 support level, Apriori took 676 milliseconds (ms) to generate frequent patterns, while FP-growth was significantly faster, taking only 216 ms. This shows that FP-growth is more efficient when handling a larger number of patterns, as would be the case with a lower support threshold.
When the support level was increased to 0.3, the performance of Apriori improved, with the time taken dropping to 262 ms. However, FP-growth still outperformed Apriori, taking just 177 ms.
At a 0.4 support level, the difference between the two algorithms narrows, but FP-growth remains the faster option, taking 182 ms compared to Apriori's 215 ms.

Q2:
For this question, I applied a minimum support threshold of 0.4.
Reasons for Choosing FP-Growth:

- Superior performance: FP-Growth consistently demonstrated faster execution times and higher memory efficiency, particularly when dealing with large datasets and lower support levels.
- Scalability: FP-Growth can efficiently handle the large dataset (bank.arff) without generating an overwhelming number of candidate itemsets, unlike the Apriori algorithm, which can become cumbersome with a large number of candidates.

Thus, FP-Growth is the best-suited algorithm for this task.
Patterns from **Bank_no** Dataset (Customers who did not subscribe):

1. Attributes:
    - default_credit: no
    - loan: no
    - marital: married
    - past_marketing: unknown
    - call_duration: 100-500 seconds

2. Top 5 Frequent Size-3 Patterns:

Pattern	Support
default_credit, loan, marital	19,799
default_credit, past_marketing, marital	20,357
default_credit, loan, call_duration	21,217
default_credit, call_duration, past_marketing	21,352
default_credit, loan, past_marketing	27,371

Patterns from **Bank_yes** Dataset (Customers who subscribed):

a. **Attributes**:

- age: 21-30 years old
- marital: married
- default_credit: no
- loan: no
- call_duration: 100-500 seconds
- housing: no

- past_marketing: unknown

b. Top 5 Frequent Size-3 Patterns:

Pattern	**Support**
marital, default_credit, loan	**2,471**
age, default_credit, loan	**2,519**
default_credit, loan, call_duration	**2,712**
default_credit, housing, loan	**3,120**
default_credit, loan, past_marketing	**2,988**

Common Characteristics:
Both subscribers and non-subscribers share the pattern of having no default credit, no loan, and a call duration of 100-500 seconds. This suggests that financial stability (e.g., not having defaults or loans) is a common characteristic across both groups.

Differences:
- Non-subscribers are more likely to be married and have unknown outcomes in previous marketing campaigns, which could indicate a lack of personalized marketing targeting these individuals effectively.
- Subscribers are typically younger (21-30 years old) or married with no housing loans. This implies that targeting younger, financially stable customers without housing loans may enhance subscription rates for the bank's products.

Q3:
Patterns from Bank_yes Dataset (Customers who subscribed):

Pattern	Support
default_credit = no, marital = married	2,735
default_credit = no, age = 21-30s	2,781
default_credit = no, loan = no, call_duration = 100-500s	2,712
default_credit = no, loan = no, housing = no	3,120
default_credit = no, loan = no, past_marketing = unknown	2,988

Patterns from Bank_no Dataset (Customers who did not subscribe):

Pattern	Support
default_credit = no, past_marketing = unknown, loan = no	27,371
default_credit = no, housing = yes	22,789
default_credit = no, past_marketing = unknown, call_duration = 100-500s	21,352
default_credit = no, loan = no, call_duration = 100-500s	21,217
default_credit = no, balance = below-1k	20,528

Similarities:
- Both subscribers and non-subscribers share the pattern of default_credit = no, loan = no, call_duration = 100-500s, indicating that customers who are financially stable (with no defaults or loans) and have had moderate phone call durations are common in both groups.
- Having no default credit is a prominent characteristic for both classes, reflecting financial stability as a key factor across the entire customer base.

Differences:
- Past Marketing Outcomes: Non-subscribers are more likely to have unknown past marketing outcomes (27,371), compared to only 2,988 for subscribers. This suggests that past marketing efforts were less successful for non-subscribers, possibly indicating a need for better-targeted marketing strategies.
- Housing Loans: Non-subscribers are more likely to have housing loans (22,789), while subscribers tend to not have housing loans (3,120). This could indicate that those without housing loans may be more inclined to subscribe.

- Financial Constraints: A significant portion of non-subscribers has lower bank balances (below-1k) (20,528), which could be a financial barrier preventing them from subscribing to additional services.
- Age: Subscribers are more likely to be younger (21-30 years old) (2,781), a trend that is not observed among non-subscribers, implying that younger customers are more likely to subscribe to the bank's term-deposit product.

These insights suggest that the bank can potentially improve its marketing efforts by focusing on younger customers without housing loans and by addressing the financial constraints of non-subscribers, such as those with low balances.

Q4:

Bank_yes Dataset (Subscribed = Yes):

Rule	Support	Confidence
past_marketing = success ==> subscribed = yes	978	0.647
default_credit = no, past_marketing = success ==> subscribed = yes	978	0.648
default_credit = no, loan = no, past_marketing = unknown, call_duration = 500-1k ==> subscribed = yes	1,061	0.358
loan = no, past_marketing = unknown, call_duration = 500-1k ==> subscribed = yes	1,079	0.359
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	1242	0.35
default_credit = no, loan = no, call_duration = 500-1k ==> subscribed = yes	1428	0.392
loan = no, call_duration = 500-1k ==> subscribed = yes	1448	0.393
default_credit = no, call_duration = 500-1k ==> subscribed = yes	1614	0.379
call_duration = 500-1k ==> subscribed = yes	1646	0.38

Bank_no Dataset (Subscribed = No):

Rule	Support	Confidence
marital = married ==> subscribed = no	24,459	0.898
default_credit = no, call_duration = 100-500s ==> subscribed = no	25,538	0.898
call_duration = 100-500s ==> subscribed = no	26,044	0.9
default_credit = no, loan = no, past_marketing = unknown ==> subscribed = no	27,371	0.901
loan = no, past_marketing = unknown ==> subscribed = no	27,814	0.901
default_credit = no, loan = no ==> subscribed = no	32,685	0.872
default_credit = no, past_marketing = unknown ==> subscribed = no	32,862	0.907
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

Common Characteristics:

- Call Duration: Both subscribers and non-subscribers share a common pattern involving the call duration. In the Bank_yes dataset, customers with a call duration between 500 and 1000 seconds tend to subscribe. Similarly, in the Bank_no dataset, shorter call durations (100-500 seconds) are prevalent among non-subscribers.
- Default Credit: Customers with no default credit appear frequently in both datasets, indicating that financial stability (no defaults) is a common trait among both groups.

Differences:

- Past Marketing: In the Bank_no dataset, unknown past marketing outcomes are strongly associated with non-subscribers, with a high confidence of 0.901, while in the Bank_yes dataset, successful past marketing has a higher confidence (0.647) for subscribers. This suggests that past successful interactions play a significant role in customer conversions, while previous unsuccessful

campaigns (or unknown outcomes) are more common among non-subscribers.

- Loan and Housing: Non-subscribers are frequently associated with no loans and unknown past marketing outcomes, while subscribers are associated with no housing loans and call durations between 500 and 1000 seconds.
- Call Duration: Non-subscribers tend to have shorter call durations (100-500 seconds), while subscribers are more likely to have longer call durations (500-1000 seconds).

Conclusion:
The analysis shows key patterns and behaviours for both subscribers and non-subscribers, highlighting factors like past marketing success and call duration that significantly influence customer behaviour. Additionally, financial stability, represented by "no default credit," is common to both groups, but other factors such as loan status and past marketing results help differentiate between customers who subscribe and those who do not.

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

A company[1] has created a large medical dataset focusing on COVID-19 infection by collectingpeople'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. One value is about infection risk and the other is about mortality risk. For this task, we are interested in infection risk only. You need to write a Java program to train some classifiers using this dataset for predicting COVID19 infection risk.

The original dataset contains about one million records with 39 attributes. For this task, we only use a subset of the data consisting of 4,612 records with a set of selected attributes. Youcan download the dataset **`COVID19.arff`** from the Canvas.

In the dataset, each record contains one person's response to the online calculator. There are 22 attributes in the dataset, which are described in the table below.

No	Column name	Description
1	Gender	Male, Female or other
2	Age	Age quantile
3	Blood_type	Type of the person's blood
4	Race	Race of the person
5	Smoking	Information on how often the person smokes
6	Public_transport_count	Number of people contacted by the person during public transportation
7	Working	Status of the person's work
8	covid19_symptoms	1 and 0 stating the existence or not
9	covid19_contact	
10	asthma	
11	kidney_disease	
12	liver_disease	
13	compromised_immune	
14	heart_disease	
15	lung_disease	
16	diabetes	
17	hiv_positive	
18	hypertension	
19	other_chronic	

20	nursing_home	
21	health_worker	
22	risk_infection	Risk of the person to get infected, high or low.

Before writing your Java program, you are required to analyse the dataset in Weka to select 3 classification algorithms based on their classification performance. For evaluation, you can use the default 10-fold cross-validation.

1. Classification in Weka

For each of the following questions, you need to

- Describe your working process in Weka.
- Provide evidence to justify your decision. The evidence can be tables to show performance comparison, screenshots, or some outputs from Weka.

1) Choose four classification algorithms from **`NaiveBayes`**, **`IBk`**, **`PART`**, **`OneR`**,ZeroR, **`J48`** based on their classification accuracy performance.

2) Choose one Evaluator from InfoGainAttributeEval or GainRatioAttributeEval and determine the number of attributes to use for each of the four chosen algorithms inorder to achieve a relatively better classification performance by that algorithm.How many attributes that you would like to use for each of the four algorithms? There are 21 attributes in this dataset (excluding the class attribute). For saving yourtime, you may decide a number between 3 to 10 attributes for each of the fouralgorithms using one of the two evaluators. This question is to determine the **number** of attributes to use for each algorithm rather than deciding what attributesto use.
   Based on the best classification performance among the four algorithms using the chosen number of attributes, choose two from the four algorithms. What are the twoalgorithms? Provide the performance comparison.

3) For this question, you need to complete a cost sensitive analysis to the two chosen algorithms from question 2). You can assume that class "infection_risk = high" is more important to you and you want to minimize the classification error to this class.Based on your analysis, which algorithm that you would like to use in order to minimize the cost? Show your cost matrix and justify your decision by providing evidence.

Question 1:
At the first step, we choose the classifications

Algorithm	**Accuracy (%)**	**Correctly Classified Instances**	**Precision (High Risk)**	**Recall (High Risk)**	**F-Measure (High Risk)**
NaiveBayes	94.23	4346	0.933	0.869	0.9
IBk	93.67	4320	0.912	0.872	0.891
OneR	95.29	4395	0.994	0.847	0.915
J48	95.86	4421	0.987	0.872	0.926
PART	95.64	4411	0.981	0.87	0.922

First, I ran each of the 6 candidate algorithms using their default settings and identified the best performing 4 to be J48, PART, OneR and NaïveBayes, with lBk's performance being slightly worse, and ZeroR being unsurprisingly worst by far.

Lazy lBk		
K	RMSE	Correct Instances
1	0.238	4320
2	0.22	4354
3	0.2136	4388
4	0.2127	4390
5	0.2114	4390
6	0.2095	4392
7	0.2094	4389
8	0.2092	4391
9	0.2083	4392
10	0.2098	4391
11	0.2106	4390
12	0.2109	4390

As there are no parameters for ZeroR to tweak to improve performance, I excluded this one. I then went about testing parameters for lBk to see if it could achieve a better performance than and of the shortlisted algorithms. K=9 had the lowest RMSE of K values up to 12 and outperformed the NaiveBayes algorithm on default parameters. I subsequently investigated parameters for NaiveBayes and found it outperformed lBk using parameter useKernelEstimator true. No further improvements could be found for the lBk algorithm, so the final selected algorithms stood as **J48, PART, OneR and NaïveBayes**.

Question 2:

NaiveBayes	
default	4346
useKernelEstimator	4403
useSupervizedDiscretization	4398

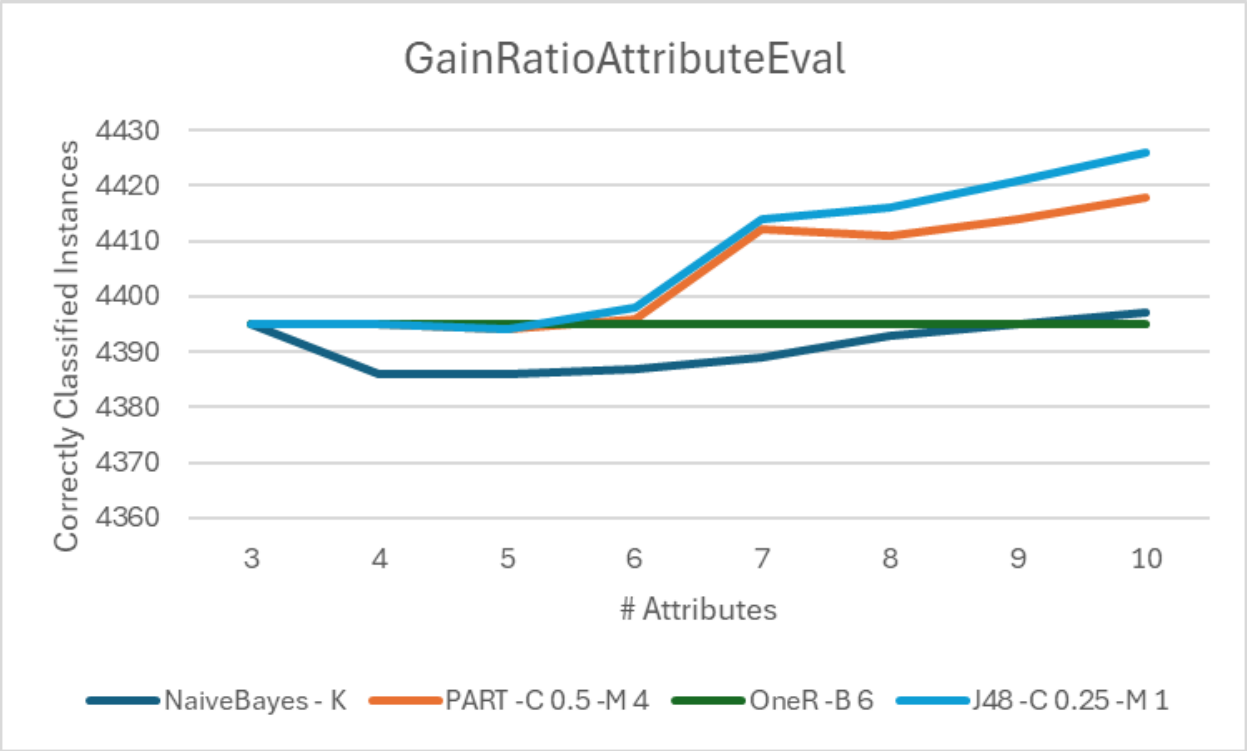
GainRatioAttributeEval	NaiveBayes - K	PART -C 0.5 -M 4	OneR -B 6	J48 -C 0.25 -M 1
3	4395	4395	4395	4395
4	4386	4395	4395	4395
5	4386	4394	4395	4394
6	4387	4396	4395	4398
7	4389	4412	4395	4414
8	4393	4411	4395	4416
9	4395	4414	4395	4421
10	4397	4418	4395	4426

NaiveBayes: has the most consistent performance. It reaches a maximum of 4397 correctly classified instances when using 10 attributes.

PART: improves significantly as the number of attributes increases, reaching a peak of 4418 correctly classified instances with 10 attributes.

OneR: is constant across all the attribute numbers with 4395 correctly classified instances, suggesting it doesn't significantly benefit from more attributes.

J48: shows the highest performance, reaching 4426 correctly classified instances with 10 attributes. It consistently improves as the number of attributes increases, which suggests that it benefits the most from having more information available.

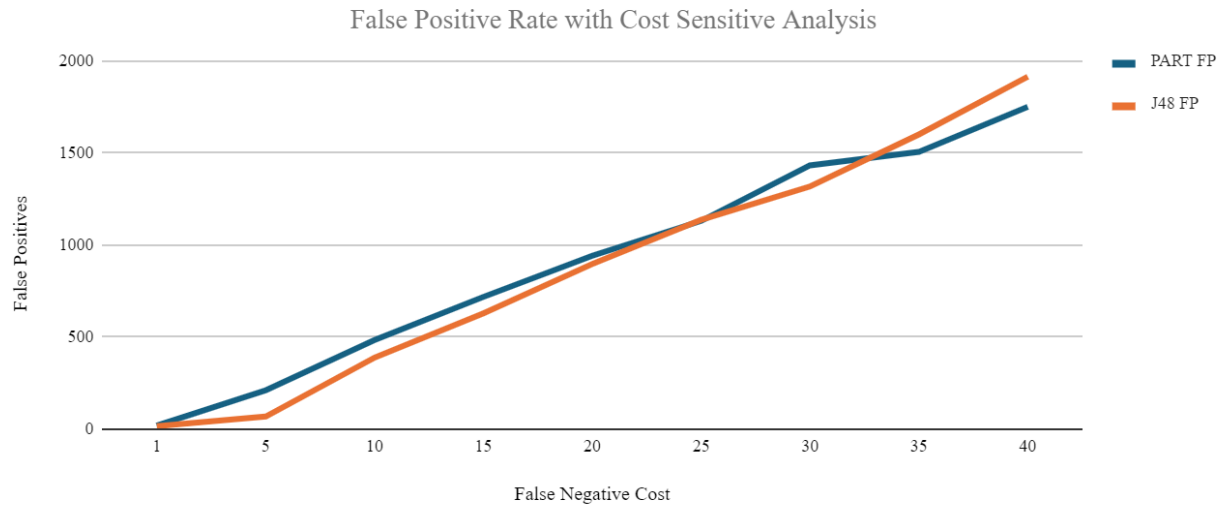


Question 3:

Cost Matrix	PART FP	PART FN	J48 FP	J48 FN	PART FP/FN Ratio	J48 FP/FN Ratio
0 1 1 0	18	174	14	174	0.1034482759	0.08045977011
0 5 1 0	209	153	67	159	1.366013072	0.4213836478
0 10 1 0	483	122	387	118	3.959016393	3.279661017
0 15 1 0	717	105	628	101	6.828571429	6.217821782
0 20 1 0	941	92	896	84	10.22826087	10.66666667
0 25 1 0	1130	81	1136	68	13.95061728	16.70588235
0 30 1 0	1432	62	1317	70	23.09677419	18.81428571
0 35 1 0	1505	64	1600	59	23.515625	27.11864407
0 40 1 0	1750	42	1914	44	41.66666667	43.5

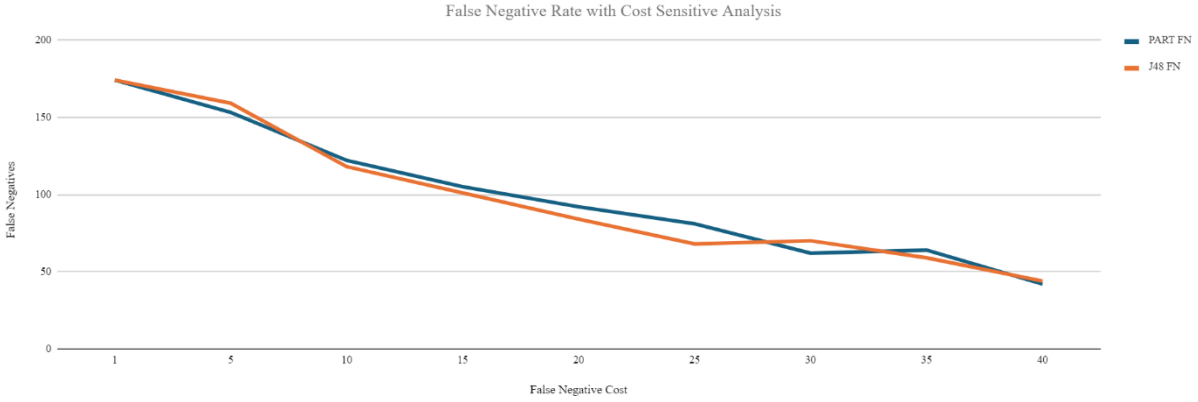
After reading this table we can understand that:

1- At lower costs (e.g., cost 1), both algorithms have similar false negatives (174), but PART has slightly more false positives.
2- As the cost increases, J48 tends to increase false positives more than PART while slightly reducing false negatives.
3- For example, at a cost of 40, J48 has 1914 false positives and 44 false negatives, whereas PART has 1750 false positives and 42 false negatives.
4- The FP/FN ratio grows more significantly for J48, showing its tendency to prioritize avoiding false negatives more aggressively than PART.



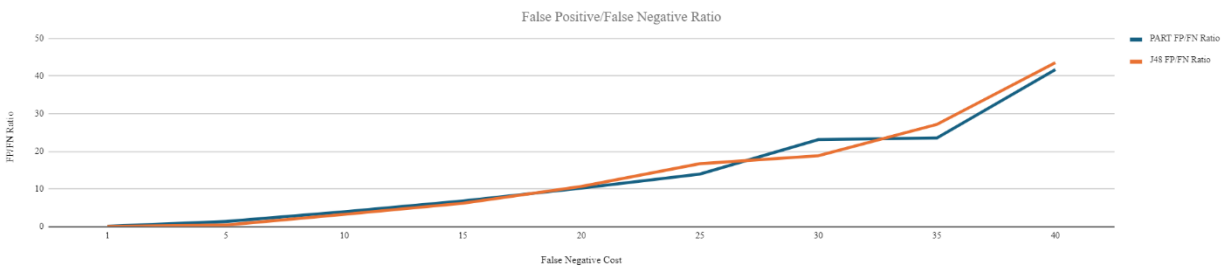
This chart compares the ratio between false positives and false negatives as the false negative cost increases. It shows:

1- The FP/FN ratio increases as the false negative cost rises, meaning as the cost of misclassifying high-risk individuals (false negatives) increases, both classifiers adjust by increasing false positives (correctly prioritizing high-risk individuals).
2- J48 tends to have a higher FP/FN ratio than PART as the cost increases, suggesting that J48 becomes more aggressive in classifying individuals as high risk to avoid false negatives.



This chart shows how false positives change as the false negative cost increases:

1- As the false negative cost increases, both PART and J48 classifiers increase the number of false positives.
2- PART starts with a lower false positive rate but eventually converges with J48 as the false negative cost rises.



This chart shows how false negatives change as the false negative cost increases:

1 Both PART and J48 reduce false negatives as the cost increases, which is expected since misclassifying high-risk individuals becomes more costly.
2 J48 starts with a slightly higher false negative rate but becomes more aggressive in reducing

false negatives as the cost rises, outperforming PART at some cost levels.

J48 appears to be more sensitive to false negative costs, aggressively reducing false negatives at the expense of increasing false positives.
PART is more balanced but may not reduce false negatives as much as J48 at higher costs.

So, J48 is the recommended classifier for minimizing false negatives in high-risk COVID-19 infection cases. While PART offers a balanced performance, J48 more effectively reduces false negatives, which is critical for ensuring high-risk individuals are not misclassified as low risk. Although J48 slightly increases false positives, this is an acceptable trade-off given the importance of prioritizing high-risk cases. Therefore, J48 is the best choice for this task.

---

2. Java program

For this part, you are required to develop a Java program to build classifiers for COVID19 infection risk prediction. Your program should satisfy the following requirements:

1) Perform the classification task using the 4 algorithms chosen in question 1.1 of thistask with the number of attributes determined in question 1.2. Your program shoulddisplay the correctly classified instances values and accuracy. The accuracy resultsshould be the same as the results obtained in question 1.2 using Weka.

2) Perform the classification task using the 2 algorithms chosen in question 1.3 of thistask 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 1.3 using Weka.

   Hint: You can use Weka class **`CostSensitiveClassifier`** (https://weka.sourceforge.io/doc.dev/weka/classifiers/meta/CostSensitiveClassifie r.html) to do the classification.

Part 1:

```
Correctly Classified Instances: 4397.0
ASC Cross Eval: 0.9533824804856895

Classifier: PART, Filter with ranker: GainRatioAttributeEval
Correctly Classified Instances: 4418.0
ASC Cross Eval: 0.9579358196010408

Classifier: OneR, Filter with ranker: GainRatioAttributeEval
Correctly Classified Instances: 4395.0
ASC Cross Eval: 0.9529488291413704

Classifier: J48, Filter with ranker: GainRatioAttributeEval
Correctly Classified Instances: 4426.0
ASC Cross Eval: 0.9596704249783174
```

For the filtered classification, the models were trained with attributes selected using the GainRatioAttributeEval evaluator and Ranker method

The results for each classifier is:

PART: Correctly classified 4418 instances with an accuracy of approximately 95.79%. This shows that PART is a robust classifier for this dataset, handling the filtered attributes well.

OneR: Correctly classified 4395 instances with an accuracy of approximately 95.29%. While OneR is typically a simpler rule-based algorithm, it still performs comparably to the more complex models like PART and J48, but with a slightly lower accuracy.

J48: This decision tree classifier correctly classified 4426 instances with an accuracy of 95.97%. This indicates that J48 performs the best out of the classifiers when using filtered attributes, handling the complexity of the dataset effectively.

Overall, the filtered classifiers are performing very well with high accuracies, ranging between 95.29% and 95.97%.

Part 2:

```
Classifier: PART

Confusion Matrix:
1357.0  14.0
2760.0  481.0

False Positives: 2760.0
False Negatives: 14.0
Total Cost: 3250.0
Precision: 0.329608938547486

Classifier: J48

Confusion Matrix:
1370.0  1.0
3202.0  39.0

False Positives: 3202.0
False Negatives: 1.0
Total Cost: 3237.0
Precision: 0.2996500437445319
```

When applying cost-sensitive classification, the goal is to minimize the misclassification of high-risk COVID-19 infection cases, particularly reducing **false negatives** (i.e., people at high risk who are incorrectly classified as low risk). Let's interpret the results for **PART** and **J48**:

- PART:
  Confusion Matrix:
  1357 true positive 14 false negatives
  2760 false positives 481 true negatives

  False Positives: 2760 False Negatives: 14

  Total Cost: 3250.0 Precision: 0.3296

The PART classifier minimizes the number of false negatives—only 14 out of 1371 high-risk individuals—but at the expense of precision through a lot of false positives. This suggested that PART favoured the classification of the high-risk individuals over the low-risk ones, which is quite appropriate for this dataset since misclassifying a high-risk individual as low risk is costlier.

- **J48**:
  Confusion Matrix:
  1370 true positives 1 false negative
  3202 false positives 39 true negatives.

False Positives: 3202	False Negatives: 1
Total Cost: 3237.0	Precision: 0.2996

For the J48 classifier, it does cut the false negatives nearly completely, but at the cost of a very large number of false positives: 3202. The overall cost is slightly smaller than PART's, meaning that J48 profoundly reduced the most expensive misclassifications—the false negatives—which was exactly the goal of performing the cost-sensitive analysis.

Comparison of PART vs. J48 (Cost-Sensitive Analysis):

False Negatives: J48 has almost no false negatives, 1, while PART has 14. This means it is more effective in picking out high-risk individuals. • False Positives: J48 has more false positives, 3202, than PART, 2760. As more low-risk are mislabelled as high-risk, precision decreases. • Total Cost: J48 has slightly lower in total cost 3237.0 against 3250.0 of PART, which is slightly more cost-effective in this setup, given that it has more false positives present.
In a nutshell, the results are that both J48 and PART would perform fairly well in this setting with a bias toward cost sensitivity, but J48 is way more aggressive toward minimizing false negatives, hence making J48 the classifier of choice where high-risk cases become the top priority.

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 parametersin 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 theattributes.

1) Which parameters in the filter that you want to tune? What are the chosen valuesfor these parameters?
2) Briefly describe your working process in Weka to determine the values for the parameters in the filter. Provide evidence to show your working.

Part 1:

TASK 3	IDFTransform	TFTransform	normalizeDocLength	stemmer	stopwordsHandler	wordsToKeep	Correct	Matrix	Accuracy
1	FALSE	FALSE	FALSE	NullStemmer	Null	100	8222	a b c d <-- classified as 3494 277 895 215 | a = computer 322 1377 642 284 | b = politics 1085 672 1791 400 | c = science 290 312 402 1560 | d = sports	59%
2	FALSE	FALSE	FALSE	LovinStemmer	Rainbow	100	10726	a b c d <-- classified as 3995 133 652 101 | a = computer 151 2011 346 117 | b = politics 770 385 2613 180 | c = science 131 123 203 2107 | d = sports	77%
3	FALSE	FALSE	TRUE	LovinStemmer	Rainbow	100	10577	a b c d <-- classified as 3939 108 714 120 | a = computer 157 1985 349 134 | b = politics 784 379 2593 192 | c = science 130 158 216 2060 | d = sports	75%
4	FALSE	TRUE	FALSE	LovinStemmer	Rainbow	100	10726	a b c d <-- classified as 3995 133 652 101 | a = computer 151 2011 346 117 | b = politics 770 385 2613 180 | c = science 131 123 203 2107 | d = sports	76.5%
5	TRUE	TRUE	FALSE	LovinStemmer	Rainbow	100	10754	a b c d <-- classified as 3987 115 679 100 | a = computer 134 2051 327 113 | b = politics 781 359 2619 189 | c = science 152 125 190 2097 | d = sports	76.7%
6	TRUE	FALSE	FALSE	LovinStemmer	Rainbow	100	10726	a b c d <-- classified as 3995 133 652 101 | a = computer 151 2011 346 117 | b = politics 770 385 2613 180 | c = science 131 123 203 2107 | d = sports	76.5%
7	TRUE	TRUE	FALSE	IteratedLovinStemmer	Rainbow	100	10799	a b c d <-- classified as 4036 114 616 115 | a = computer 162 2016 335 112 | b = politics 767 360 2663 158 | c = science 144 151 185 2084 | d = sports	77%
8	TRUE	TRUE	FALSE	SnowballStemmer	Rainbow	100	10625	a b c d <-- classified as 4012 104 601 164 | a = computer 120 2065 300 140 | b = politics 767 292 2619 270 | c = science 194 163 278 1929 | d = sports	75.7%

Based on the results, these parameters has been considered as important for turning:

		Chosen Value
IDFTransform	setting IDFTransform = TRUE generally improves performance. Best performance was seen in configurations where IDF was applied, so it's likely important.	TRUE
TFTransform	applying TF (TFTransform = TRUE) resulted in better performance. Configurations with both TF and IDF enabled generally had higher accuracy.	TRUE
normalizeDocLength	document length normalization (normalizeDocLength = TRUE) was applied only in Task 3, and the results showed a slight decrease in performance. Hence, it's likely less useful in this scenario.	FALSE
stopwordsHandler	the **Rainbow** stopwords handler was applied and performed consistently well, suggesting that removing stopwords improves classification performance.	Rainbow
Stemmer	The table shows that using stemming improves performance significantly over not using a stemmer.	IteratedLovinStemmer

---

Part 2:

I selected the String to Word Vectorizer filter in the Weka and ran the J48 classifier with the default filter parameters, after making sure to set Words to Keep to 100 and '@@class@@' as attribute, as these would be constant across all tests. Following this, I set stopwords and stemmer to Rainbow and LovinStemmer, respectively, as these are standard options, ran the classification again. From this, I experimented with IDFTransform, TFTransform, normalizeDocLength, & stemmer, and kept track of the results. Between each test I undid my previous alterations to the filter parameters.

---

2. Java program

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

```
Correctly Classified Instances       10948               78.0996 %
Incorrectly Classified Instances      3070               21.9004 %
Kappa statistic                          0.7002
Mean absolute error                      0.1371
Root mean squared error                  0.2918
Relative absolute error                 37.527  %
Root relative squared error             68.2609 %
Total Number of Instances            14018

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.797    0.084    0.836      0.797    0.816      0.721    0.920     0.856     computer
               0.780    0.034    0.839      0.780    0.808      0.767    0.915     0.792     politics
               0.730    0.151    0.655      0.730    0.690      0.561    0.855     0.687     science
               0.832    0.035    0.843      0.832    0.837      0.801    0.934     0.835     sports
Weighted Avg.  0.781    0.084    0.787      0.781    0.783      0.699    0.903     0.793

===== Evaluating on filtered (training) dataset done =====
Executing J48: 168.3271957 seconds

Done

Correctly Classified Instances       10507               74.9536 %
Incorrectly Classified Instances      3511               25.0464 %
Kappa statistic                          0.6533
Mean absolute error                      0.1273
Root mean squared error                  0.349
Relative absolute error                 34.8202 %
Root relative squared error             81.6341 %
Total Number of Instances            14018

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.937    0.225    0.690      0.937    0.795      0.680    0.890     0.766     computer
               0.781    0.045    0.800      0.781    0.790      0.743    0.925     0.786     politics
               0.447    0.047    0.789      0.447    0.571      0.492    0.813     0.646     science
               0.826    0.042    0.816      0.826    0.821      0.781    0.938     0.840     sports
Weighted Avg.  0.750    0.107    0.762      0.750    0.736      0.657    0.884     0.749

===== Evaluating on filtered (training) dataset done =====
Executing HoeffdingTree: 66.9147346 seconds

Done

Correctly Classified Instances       10708               76.3875 %
Incorrectly Classified Instances      3310               23.6125 %
Kappa statistic                          0.6768
Mean absolute error                      0.1209
Root mean squared error                  0.3334
Relative absolute error                 33.0855 %
Root relative squared error             77.994  %
Total Number of Instances            14018

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.772    0.077    0.842      0.772    0.806      0.710    0.866     0.763     computer
               0.763    0.048    0.787      0.763    0.775      0.724    0.863     0.666     politics
               0.748    0.170    0.633      0.748    0.686      0.552    0.799     0.578     science
               0.774    0.030    0.850      0.774    0.810      0.772    0.882     0.724     sports
Weighted Avg.  0.764    0.089    0.774      0.764    0.767      0.679    0.850     0.686

===== Evaluating on filtered (training) dataset done =====
Executing IBk: 76.5013097 seconds

Done

Correctly Classified Instances       11327               80.8033 %
Incorrectly Classified Instances      2691               19.1967 %
Kappa statistic                          0.7364
Mean absolute error                      0.2719
Root mean squared error                  0.3449
Relative absolute error                 74.3909 %
Root relative squared error             80.6913 %
Total Number of Instances            14018

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.820    0.069    0.863      0.820    0.841      0.760    0.928     0.816     computer
               0.790    0.025    0.878      0.790    0.832      0.797    0.925     0.783     politics
               0.801    0.159    0.664      0.801    0.726      0.609    0.828     0.592     science
               0.816    0.015    0.926      0.816    0.867      0.842    0.934     0.828     sports
Weighted Avg.  0.808    0.076    0.821      0.808    0.812      0.739    0.900     0.749

===== Evaluating on filtered (training) dataset done =====
Executing SMO: 107.4755595 seconds
```

PART 1:

1. **IBk and SMO** have the highest accuracy (80.80%). They perform well across all classes, particularly "Computer" and "Sports," but struggle somewhat with "Science."
2. **J48** has a moderate accuracy of 77.55%, but it struggles significantly with "Science," where many instances are misclassified.
3. **HoeffdingTree** achieves 76.39% accuracy, similar to J48. It also faces challenges with "Science" but is faster in processing time, making it more suitable for real-time tasks.

Overall, **IBk and SMO** are the best for accuracy, while **HoeffdingTree** excels in time efficiency.

PART 2:

In this table I'm giving a very organized explanation to the question:

Classifier	**Correctly Classified Instances**	**Accuracy**	**Time Taken**
J48	10,871	77.55%	168.3271957 seconds
HoeffdingTree	10,708	76.39%	66.9147346 seconds
IBk	11,327	80.80%	76.5013097 seconds
SMO	11,327	80.80%	107.4755595 seconds

PART 3:

HoeffdingTree performs the best in terms of time efficiency, taking **66.9147 seconds** to complete the task.

Why is HoeffdingTree faster than other algorithms?

1. **Incremental Learning**: HoeffdingTree uses an incremental learning approach, meaning it can make decisions (splits) based on smaller subsets of data, allowing it to process data and make predictions faster compared to other algorithms like J48, which constructs the entire decision tree at once.
2. **Memory Efficiency**: HoeffdingTree is designed to handle streaming data efficiently, and it doesn't need to keep all data in memory at once. This makes it faster than algorithms like IBk (which requires calculating distances for all instances) and SMO (which optimizes a large margin classifier that involves quadratic programming, a computationally intensive process).
3. **Simpler Structure**: Compared to the other classifiers like J48 and IBk, HoeffdingTree builds a simpler tree structure by evaluating fewer splits based on the Hoeffding bound, which leads to faster decision-making.

In contrast, algorithms like IBk and SMO involve more computationally intensive tasks such as calculating distances between instances (in the case of IBk) or solving optimization problems (in the case of SMO). Therefore, HoeffdingTree achieves better time efficiency.

Submission Requirements

1. Your report should include your answers to the questions described above. It shouldbe structured well and easy to read. Your answers should be justified by evidence.
2. Your Java programs should be commented appropriately and should minimize repeated code.
3. Your work must be completed by you own or your team.
4. Submit one single zip file which contains your report and three Java projects.
5. If you are in a team, select “People” on Canvas to choose one of the groups to registeryour team. Your team should submit one submission. Your submission file name should be: **student1ID_student2ID.zip**.
6. If you complete the assignment individually, your submission file name should be: **studentID.zip**.

General Marking Scheme

Task 1 (12 marks)	
Questions Clearly describe the approach for generating answers for each question Provide evidence to support your decision	6
Java program Correctly accomplish the required tasks using the right algorithms Produce the required outputs Coding quality	6
Task 2 (12 marks)	
Data analysis in Weka Clearly describe the approach for generating answers for each question Provide evidence to support your decision	6
Java program Correctly accomplish the required tasks using the right algorithms Produce the required outputs Coding quality	6
Task 3 (10 marks)	
Attribute selection in Weka Clearly describe your approach for generating answers for each question Provide evidence to support your decision	5
Java program Correctly accomplish the required tasks using the right algorithms Produce the required outputs Coding quality	5
Report presentation (1 mark)	