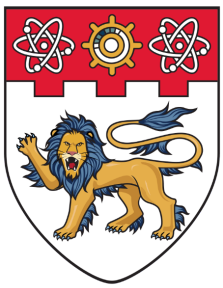

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4032 - Data Analytics and Mining

Project 1

Team Members (Group 27):

Arya Shashwat (U1822436J)

Singh Kirath (U1922329J)

Chatterjee Pramurta (U1822597G)

Hasan Mohammad Yusuf (U1822478E)

1. Implementation of a classifier based on association rules

1.1. Pre-Processing:

- **Numerical Data:** An entropy-based discretization is performed on the numerical data where for each potential split, the entropy and entropy gain are calculated, and the split with the highest entropy gain is chosen. Then, we recursively perform the partition on each split until we reach the base condition where the entropy gain is negative or zero.

- **Categorical Data:** Label Encoding is performed on columns having categorical values.

- **Missing Values:** The missing values in a column are replaced with the most frequent value in that particular column.

For the scope of this task, we used the CBA-RG algorithm (from the KDD-98 paper) for rule generation and the M1 algorithm for classification.

1.2. Mining Class Association Rules (CARs):

Association rule mining on a dataset D with attributes I produces a list of all association rule items R such that if $r \in R$, then r has the format $(x1) \rightarrow (x2)$ where $x1, x2 \in I$. This set of R association rules can be used to identify relationships between data items and make predictions. Apriori algorithm is one such technique to mine association rules from a dataset.

Classification association rules (CARs) are a subset of R ($C \subseteq R$) such that each ruleitem $c \in C$ has the format $(x) \rightarrow y$, where $x \in I$ and y is a unique class label. This set of CARs represent a relationship between the data attributes and their respective class labels which can be used to build a classifier.

CBA-RG has been utilised in this task for the generation of CARs **[1]**. This algorithm performs multiple passes over the dataset until convergence to generate the final list of CARs.
The kth pass of this algorithm can be broken down into the following 4 key steps-

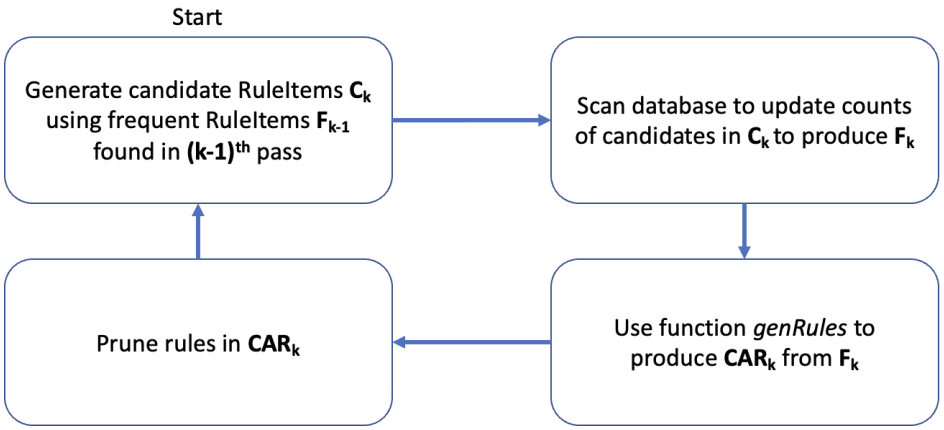

Fig. 1 - Key steps in CBA-RG

After the above 4 steps, the algorithm finally converges to produce a set of CARs that can be used to build a classifier.

1.3. Building the M1 classifier:

Ideally, to build the best classifier we would have to check the error rate of each subset of the CARs against the training data and select the one that gives the least error. But this approach is infeasible as there would be a total of 2^m (m = total rules items) such subsets and the calculation would take an unprecedented amount of time.

The M1 algorithm is a heuristic approach to this problem and does not require the computation of all the subsets [1]. It takes in the CARs generated by the BCA-RG to build a classifier. We have borrowed some code for data processing and rule generation from a github repo **[4]**.
Below is the explanation of the pseudocode for the M1 classifier-

```
Step 1  1   R = sort(R);
        2   for each rule r ∈ R in sequence do
        3       temp = ∅;
        4       for each case d ∈ D do
        5           if d satisfies the conditions of r then
        6               store d.id in temp and mark r if it correctly
                            classifies d;
Step 2  7       if r is marked then
        8           insert r at the end of C;
        9           delete all the cases with the ids in temp from D;
        10          selecting a default class for the current C;
        11          compute the total number of errors of C;
        12      end
        13  end
        14  Find the first rule p in C with the lowest total number
                of errors and drop all the rules after p in C;
Step 3  15  Add the default class associated with p to end of C,
                and return C (our classifier).
```

Fig. 2 - M1 Classifier pseudocode

Step 1 : Given a list of CARs, this step sorts them according to the “>” condition. This ensures that we only use the highest precedence rules to build our classifier. The “>” condition is defined as-

Given 2 rules R_i and R_j, ($R_i > R_j$ i.e. R_i has precedence over R_j) if:

1. R_i has higher confidence than R_j or
2. If their confidences are same, R_i has higher support than R_j or
3. If their supports are same, R_i was generated earlier than R_j

Step 2 : This step goes from lines 2-13 and is responsible for selecting the final list of rules that goes in our classifier and a default class for the cases in the dataset not covered by the list of rules.

It starts by taking a rule r from the list of rules and finds all data items $d \in D$ that are satisfied by this rule. If there is at least one such data item found, then we mark r (line 5) and add it as a potential rule in our classifier (line 8). All the data items $d \in D$ satisfied by this rule are removed from the dataset as they are already linked to a rule (line 9). A default class is also selected which is the majority class in the remaining items in the dataset (line 10). After this, the error is calculated (line 11) for our current classifier which is:

Error = *Errors by default class in training data + Errors made by selecting rules in C*

Step 3 : In the last step, we find the cutoff rule which is the first rule that records the least number of errors on the dataset. All the rules after the cutoff rule are discarded. This is done due to the fact that the rules after the cutoff only further add to the inaccuracy of the classifier and hence, discarding them increases the overall accuracy.

All the remaining undiscarded rules and the default class (from the last rule in C) form our final M1 classifier.

2. Evaluation of M1 Classifier

We have used seven datasets from the UCI machine learning portal to test the performance of our classifier **[3]**. The first five datasets are present in the KDD'98 paper. These datasets are:

1. **Iris:** R.A. Fisher's renowned 1936 study, The Use of Multiple Measurements in Taxonomic Problems, utilised the Iris dataset. Each row of the data shows one iris flower, along with the species and centimeter dimensions of its botanical parts, the sepal and petal.

2. **Zoo:** This dataset contains descriptions of various animals from a zoo. There are 16 features with various traits to describe the animals. Most features are Boolen(1 or 0), indicating the presence or absence of certain traits like hair, fins, backbone, leg etc. The 7 Class Types are: Mammal, Bird, Reptile, Fish, Amphibian, Bug and Invertebrate.

3. **Tic-Tac-Toe:** This database encodes the complete set of possible board configurations at the end of tic-tac-toe games, where "x" is assumed to have played first. The target concept is "win for x" (i.e., true when "x" has one of 8 possible ways to create a "three-in-a-row"). First nine attributes represent nine fields on the tic-tac-toe board and the tenth is a class attribute which contains information if the 'x' player won.

4. **German Credit Risk:** The original dataset contains 1000 entries with 20 categorial/symbolic attributes prepared by Prof. Hofmann. In this dataset, each entry represents a person who takes a credit by a bank. Each person is classified as good or bad credit risks according to the set of attributes like housing, property, job etc. This dataset comes encoded for categorical variables.

5. **Horse Colic:** This dataset contains information about whether or not a horse can survive based upon its past medical conditions. The dataset has 23 features and has a good mix of categorical and continuous features. The data has the horses' medical conditions like age, pulse, respiratory rate, rectal temp etc and whether the horse survived, died, euthanized etc.

6. **Vertebral Column:** This biomedical dataset was built by Dr. Henrique da Mota. Each patient is represented in the data set by six biomechanical attributes derived from the shape and orientation of the pelvis and lumbar spine (in this order): pelvic incidence, pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius and grade of spondylolisthesis. The following convention is used for the class labels: Normal (NO) and Abnormal (AB).

7. **Blood transfusion Service Center:** This data is taken from the Blood Transfusion Service Center in Hsin-Chu City in Taiwan. The data contains donation details of 748 donors. These 748 donor data, each one included R (Recency - months since last donation), F (Frequency - total number of donation), M (Monetary - total blood donated in c.c.), T (Time - months since first donation), and a binary variable representing whether he/she donated blood in March 2007 (1 stands for donating blood; 0 stands for not donating blood).

Datasets	Accuracy	No. of CARs	Runtime(RG) (sec)	Runtime(CB) (sec)	No. of Rules
Iris	92.66	121	0.053	0.001	9
Zoo	91.09	1509	0.812	0.015	10
Tic-Tac-Toe	76.26	588	1.326	0.108	58
German	70.00	1101	2.176	0.295	91
Horse	66.67	1136	0.846	0.061	52
Vertebra	68.06	522	6.204	0.026	15
Transfusion	76.14	80	0.219	0.011	11

- **Accuracy:** This is the error metric to check the performance of the classifier.
- **No. of CARs:** It gives the average numbers of rules generated by algorithm CBA-RG in each cross-validation
- **Runtime(RG) (sec):** It gives the average time taken to generate the rules in each cross-validation.
- **Runtime(CB) (sec):** It shows the average times taken to build each classifier.
- **No. of Rules:** It gives the average number of rules in the classifier built by CBA-CB.

3. Evaluation of classic ML models

We used the same datasets to test the performance of the classic machine learning models. The models used are decision tree, random forest and SVM. We used accuracy as a classification error metric to compare the performance of the above stated models. We also compared these three models to our M1 classifier. The results are given below in the table.

Datasets	Decision Tree	Random Forest	SVM	M1
Iris	93.34	94.66	94.66	92.66
Zoo	94.15	96.07	96.07	91.09
Tic-Tac-Toe	85.70	88.63	86.54	76.26
German	70.10	75.10	74.70	70.00
Horse	66.02	73.31	67.00	66.67
Vertebra	84.53	83.89	84.21	68.06
Transfusion	76.20	76.20	76.20	76.14

Observation: Apart from few datasets like Tic-Tac-Toe and Vertebra, the M1 Classifier performs fairly well on the rest of the datasets when compared with the classic models with an approx accuracy difference of 5%.

4. Improvement of the M1 Classifier

For the scope of this task, we decided to implement the Active Pruning Rules (APR) algorithm which is a modified version of the M1 algorithm **[2]**. Although the main steps taken in APR are similar to M1, there are still some significant differences in their implementation logic. Due to the rule selection process in APR, the number of rules (APR) <= rules (M1) always. The key differences between the two are as follows-

	M1 Algorithm	**APR Algorithm**
1.	Uses a horizontal rule mining algorithm like Apriori.	Uses a vertical rule mining algorithm.
2.	When selecting rules from the sorted list to add to the classifier, the data items satisfied by a particular rule are removed from the dataset and the rule is marked to be added to the classifier.	When evaluating the rules for the classifier against the dataset, all the rules except the first satisfactory rule lose out on the given data item. This leads to a biased distribution of supports among the rules, as the rules with lower ranks miss out on data items that might be satisfied by them. This is the key problem APR addresses. Upon the deletion of a data item, the ranks of all the rules that satisfy it are updated, to ensure a fair support for all the rules.
3.	After the classifier has been built, the prediction is made based on the first rule that satisfies the test data. If no such rule is found, then the prediction is made based on the default class.	Unlike the M1 algorithm, more than one rule is taken into account when making a prediction on the test data. Upon receiving a test data, all the classifier rules are selected that satisfy the test data and divided into clusters based on their class label. The class of the cluster with the highest number of rules is then assigned to the test data. Lastly, when no such rules are found, the default class is used.

The pseudocode given below explains the functioning of the APR rule evaluation logic:

```
1   The set of extracted Rules_set and the training data set (T)

2   classifier (C)

3   Temp' = rank (Rules_set)

4   θ← Classifier
5   θ← Data Structure (D_S)
6   for each training example t in T do
7   find the first ranked rule ri∈ Temp' that its attributes is inside t
8   if no rules match t then
9   keep t uncovered;
10  else

11  begin

12  D_S ← D_S ∪ ri
13  remove t
14  update the rank of rules in Temp'

15  end

16  end if

17  end for

18  discard all rules in Temp'

19  C ← C ∪ D_S
20  if |T|>0 – size of the training data set then
21  create a default rule from unclassified examples T
22  else

23  create a default rule from the current C (most class appearing with rules) and add it to C
24  end if
```

Fig. 3 - APR rule evaluation pseudocode

lines 1-3 : Initialization of the classifier (C), and the set of rules (temp).

lines 4-17 : This code checks each training pattern in the dataset and looks for rules in the rules set that satisfy the pattern. For all the rules found, it adds them to the data structure D_S, updates the ranks of all rules in temp and removes the pattern from the dataset. If no rules are found, then the data pattern is kept uncovered.

lines 18-19 : All the unused rules in temp are discarded and the classifier is built by adding all the rules in D_S.

lines 20-24 : This code block deals with the formation of the default class. If there are data patterns left in the dataset that do not satisfy any rule, then the default class rule is created using these unclassified patterns. If there are no such patterns left, then the rules in the current classifier (C) are used to generate the default class.

4.1 Comparison between the M1 and APR algorithm

Datasets	Accuracy		No. of CARs		Runtime(RG) (sec)		Runtime(CB) (sec)		No. of Rules	
	M1	APR	M1	APR	M1	APR	M1	APR	M1	APR
Iris	92.66	95.33	121	120	0.053	0.095	0.001	0.000	9	9
Zoo	91.09	96.00	1509	1452	0.812	1.711	0.015	0.008	10	9
Tic-Tac-Toe	76.26	85.49	588	564	1.326	3.221	0.108	0.088	58	33
German	70.00	72.00	1101	1014	2.176	10.546	0.295	0.221	91	36
Horse	66.67	67.00	1136	1173	0.846	1.964	0.061	0.053	52	39
Vertebra	68.06	83.22	522	522	6.204	11.410	0.026	0.013	15	13
Transfusion	76.14	78.07	80	80	0.219	0.039	0.011	0.005	11	11

Observation:

1. **Accuracy:** We can see that the APR algorithm has a better performance compared to the M1 algorithm.
2. **Runtime(CB) (sec):** We can see that the APR algorithm has a lower runtime compared to M1 algorithm.
3. **No. of Rules:** We can see that the APR algorithm generated lesser rules compared to the M1 algorithm.

5. References:

1. Bing Liu, Wynne Hsu, Yiming Ma, “Integrating Classification and Association Rule Mining,” in AAAI, 1998

2. K. D. Rajab, "New Associative Classification Method Based on Rule Pruning for Classification of Datasets," in IEEE Access, vol. 7, pp. 157783-157795, 2019, doi: 10.1109/ACCESS.2019.2950374.

3. “UCI Machine Learning Repository: Data Sets”, Archive.ics.uci.edu, available: https://archive.ics.uci.edu/ml/datasets.php

4. Liulizhi1996, “Classification Based Association”, (2018), GitHub repository, available: github.com/liulizhi1996/CBA