### Practical 1

### Introduction

Data mining, a subfield of artificial intelligence and machine learning, involves the extraction of useful information from large datasets. This process involves techniques like pattern recognition, statistical analysis, and machine learning algorithms to uncover hidden patterns, trends, and relationships within the data. Data mining applications have become increasingly prevalent across various industries, from healthcare and finance to marketing and retail.

This report provides a detailed overview of several prominent data mining applications, including time-series data mining, social network mining, recommendation systems, web mining, and text mining. Each application is explored in depth, highlighting its key techniques and potential benefits.

### Mining Time-Series Data

Time-series data, consisting of a sequence of data points collected over time, is a valuable resource for understanding trends, forecasting future events, and detecting anomalies.

**Key applications:**

* **Financial forecasting:** Predicting stock prices, exchange rates, and other financial indicators.
* **Weather prediction:** Forecasting weather patterns and climate change.
* **Medical diagnosis:** Identifying disease patterns and predicting patient outcomes.
* **Customer behavior analysis:** Understanding customer purchasing habits and preferences.

**Techniques:**

* **ARIMA models:** Autoregressive Integrated Moving Average models are widely used for time-series forecasting. They capture the dependence of the current value on past values and errors.
* **Neural networks:** Deep learning models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks are effective for forecasting complex time-series patterns.
* **Exponential smoothing:** A family of techniques that smooths out fluctuations in time-series data to reveal underlying trends.
* **Decomposition:** Breaking down time-series data into components such as trend, seasonality, and noise to better understand its structure.

### Social Network Mining

Social networks have become an integral part of modern society, and mining them can reveal valuable insights into human behavior, relationships, and communities.

**Key applications:**

* **Community detection:** Identifying groups of people with strong connections.
* **Link prediction:** Predicting future connections between individuals.
* **Influence analysis:** Identifying influential individuals in a network.
* **Sentiment analysis:** Analyzing the sentiment expressed in social media posts.

**Techniques:**

* **Graph mining algorithms:** Algorithms specifically designed for analyzing graph-structured data, such as community detection algorithms and link prediction algorithms.
* **Clustering algorithms:** Grouping similar nodes together based on their attributes and relationships.
* **Centrality measures:** Quantifying the importance of nodes in a network, such as degree centrality, betweenness centrality, and eigenvector centrality.
* **Text mining techniques:** Methods for extracting information from text data, such as sentiment analysis and topic odelling.

### Recommendation Systems in Retail

Recommendation systems have revolutionized the retail industry by providing personalized product suggestions to customers.

**Key applications:**

* **E-commerce:** Suggesting products to customers based on their past purchases and browsing history.
* **Content streaming:** Recommending movies, TV shows, or music based on user preferences.
* **Social media:** Suggesting friends, pages, or groups to follow.

**Techniques:**

* **Collaborative filtering:** Recommending items based on similarities between users or items.
* **Content-based filtering:** Recommending items based on their attributes and the user’s preferences.
* **Hybrid systems:** Combining collaborative filtering and content-based filtering to leverage the strengths of both approaches.

### Web Mining

Web mining involves extracting useful information from the World Wide Web.

**Key applications:**

* **Web search engines:** Ranking web pages based on their relevance to search queries.
* **Market research:** Understanding customer behavior and preferences.
* **Customer behavior analysis:** Analyzing user interactions with websites to improve user experience.

**Techniques:**

* **Text mining:** Extracting information from text content, such as keywords, entities, and sentiments.
* **Graph mining:** Analyzing the structure of the web, such as hyperlinks and page relationships.
* **Data mining algorithms for web logs:** Analyzing web server logs to understand user behavior and website performance.

### Text Mining

Text mining, also known as natural language processing, focuses on extracting meaningful information from unstructured text data.

**Key applications:**

* **Sentiment analysis:** Analyzing the sentiment expressed in text, such as positive, negative, or neutral.
* **Topic odelling:** Identifying the main topics discussed in a document.
* **Information extraction:** Extracting specific pieces of information from text, such as names, dates, and locations.
* **Text classification:** Categorizing text documents into predefined categories.

**Techniques:**

* **Natural language processing techniques:** Techniques for understanding and analyzing human language, such as tokenization, stemming, and part-of-speech tagging.
* **Machine learning algorithms:** Algorithms for training models to perform tasks like sentiment analysis and text classification.
* **Information retrieval techniques:** Techniques for searching and retrieving relevant information from large text corpora.

### Conclusion

Data mining has emerged as a powerful tool for extracting valuable insights from large datasets. The applications discussed in this report represent just a few examples of how data mining can be used to address a wide range of challenges and opportunities. As data volumes continue to grow, the importance of data mining will only increase.

### References

* Han, J., Kamber, M., & Pei, J. (2011). Data mining: concepts and techniques. Morgan Kaufmann.
* Witten, I. H., Frank, E., & Hall, M. A. (2016). Data mining: practical machine learning tools and techniques. Morgan Kaufmann.
* Aggarwal, C. C. (2014). Social network analysis. Springer.
* Tan, P.-N., Steinbach, M., & Kumar, V. (2006). Introduction to data mining. Addison-Wesley.
* Jurafsky, D., & Martin, J. H. (2009). Speech and language processing. Prentice Hall.

### Practical 2

Suppose that a data warehouse for Big University consists of the following four

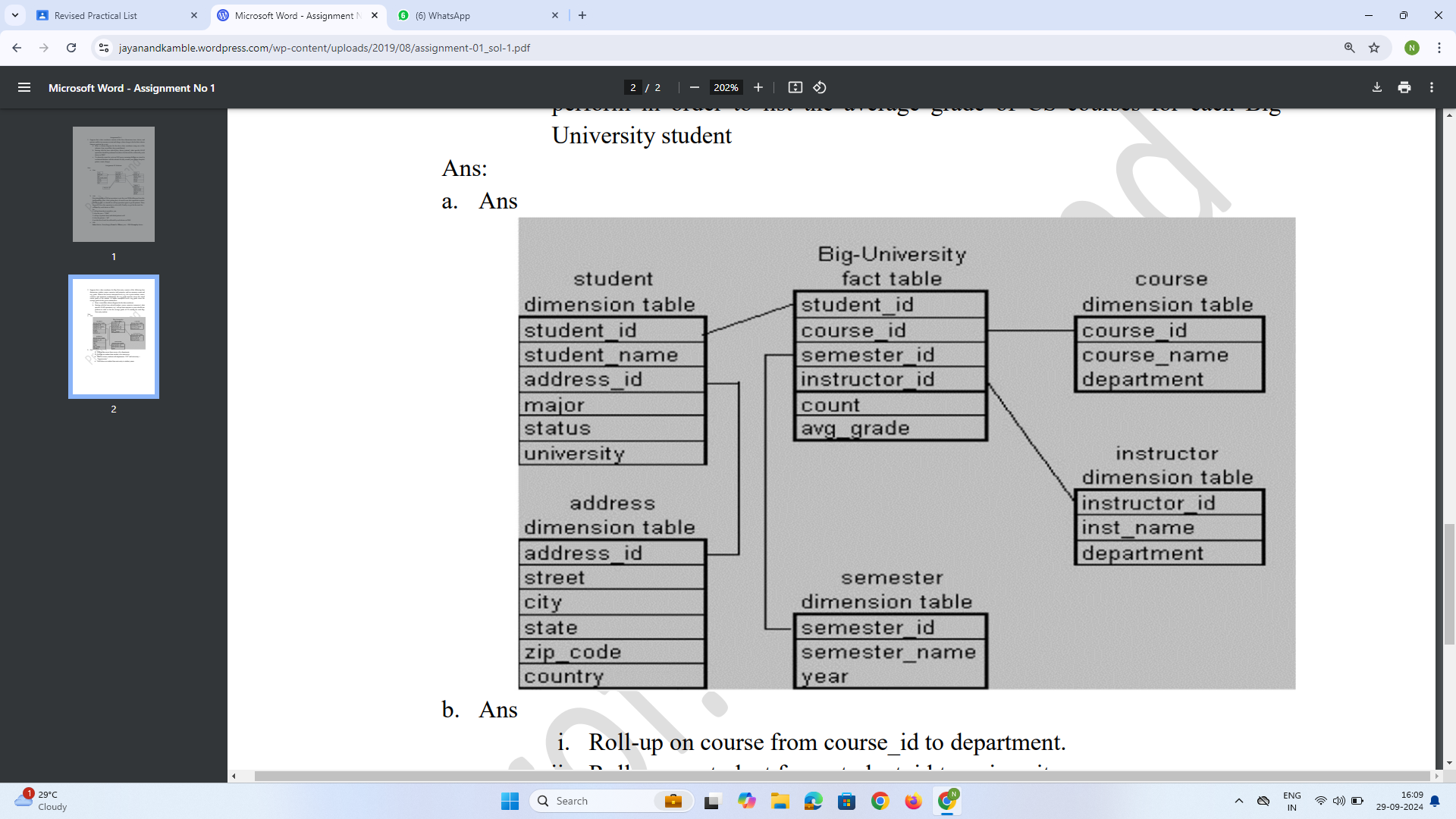
dimensions: ‘student’, ‘course’, ‘semester’, and ‘instructor’. Further, assume there are two measures ‘count’ and ‘avg\_grade’. The measure ‘count’ refers to number of students. When at the lowest conceptual level (e.g. for a given student, course, semester, and instructor combination), the ‘avg\_grade’ measure stores the actual course grade of the student. At higher conceptual levels, ‘avg\_grade’ stores the average grade for the given combination.

(a) Draw a snowflake schema diagram for the data warehouse.

(b) Starting with the base cuboid [student; course; semester; instructor], what specific OLAP operations (e.g., roll-up from semester to year) should one perform in order to list the average grade of CS courses for each Big University student.

(c) If each dimension has five levels (including all), such as “student < major < status < university < all", how many cuboids will this cube contain (including the base and apex cuboids)?

a)



b)

i. Roll-up on course from course\_id to department.

ii. Roll-up on student from student\_id to university.

iii. Dice on course, student with department ="CS" and

university = "biguniversity"

iv. Drill-down on student from university to student\_name.

c)

Li = 5-1 =4

N=4 dimensions

So, this cube will contain (Li+1)^4 = 625 cuboids

### Practical 3

Implement routines to normalize the data in the sample data file using:

a) Min-Max normalization : (i) Map to range [0,1]

(ii) Map to range [-1,1]

b) Z-score normalization

c) Decimal scaled normalization

Sample Dataset : 10, 12, 3, 6, 5, 25, 17, 100, 1000, 98, 11, 27, 78, 33, 9, 18, 23,

44, 690, 200

#include <stdio.h>

#include <math.h>

#define SIZE 20

double data[SIZE] = {10, 12, 3, 6, 5, 25, 17, 100, 1000, 98, 11, 27, 78, 33, 9, 18, 23, 44, 690, 200};

void min\_max\_normalization(double data[], double norm\_data[], int size) {

double min = data[0], max = data[0];

int i;

for (i = 1; i < size; i++) {

if (data[i] < min) min = data[i];

if (data[i] > max) max = data[i];

}

for (i = 0; i < size; i++) {

norm\_data[i] = (data[i] - min) / (max - min);

}

}

void min\_max\_normalization\_neg1\_to\_1(double data[], double norm\_data[], int size) {

double min = data[0], max = data[0];

int i;

for (i = 1; i < size; i++) {

if (data[i] < min) min = data[i];

if (data[i] > max) max = data[i];

}

for (i = 0; i < size; i++) {

norm\_data[i] = 2 \* ((data[i] - min) / (max - min)) - 1;

}

}

void z\_score\_normalization(double data[], double norm\_data[], int size) {

double sum = 0, mean, std\_dev = 0;

int i;

for (i = 0; i < size; i++) sum += data[i];

mean = sum / size;

for (i = 0; i < size; i++) std\_dev += pow(data[i] - mean, 2);

std\_dev = sqrt(std\_dev / size);

for (i = 0; i < size; i++) norm\_data[i] = (data[i] - mean) / std\_dev;

}

void decimal\_scaled\_normalization(double data[], double norm\_data[], int size) {

double max\_val = fabs(data[0]);

int i, j;

for (i = 1; i < size; i++) {

if (fabs(data[i]) > max\_val) max\_val = fabs(data[i]);

}

j = ceil(log10(max\_val));

for (i = 0; i < size; i++) {

norm\_data[i] = data[i] / pow(10, j);

}

}

void print\_array(double array[], int size) {

int i;

for (i = 0; i < size; i++) {

printf("%f ", array[i]);

}

printf("\n");

}

int main() {

double min\_max\_norm\_0\_1[SIZE], min\_max\_norm\_neg1\_1[SIZE];

double z\_score\_norm[SIZE], decimal\_scaled\_norm[SIZE];

int size = SIZE;

min\_max\_normalization(data, min\_max\_norm\_0\_1, size);

min\_max\_normalization\_neg1\_to\_1(data, min\_max\_norm\_neg1\_1, size);

z\_score\_normalization(data, z\_score\_norm, size);

decimal\_scaled\_normalization(data, decimal\_scaled\_norm, size);

printf("Original Data:\n");

print\_array(data, size);

printf("Min-Max Normalization (0,1):\n");

print\_array(min\_max\_norm\_0\_1, size);

printf("Min-Max Normalization (-1,1):\n");

print\_array(min\_max\_norm\_neg1\_1, size);

printf("Z-Score Normalization:\n");

print\_array(z\_score\_norm, size);

printf("Decimal Scaled Normalization:\n");

print\_array(decimal\_scaled\_norm, size);

return 0;

}

**Output :**

**Original Data:**

**10.000000 12.000000 3.000000 6.000000 5.000000 25.000000 17.000000 100.000000 1000.000000 98.000000 11.000000 27.000000 78.000000 33.000000 9.000000 18.000000 23.000000 44.000000 690.000000 200.000000**

**Min-Max Normalization (0,1):**

**0.007049 0.009161 0.000000 0.002792 0.001860 0.022727 0.013372 0.097674 1.000000 0.095720 0.008105 0.025581 0.075581 0.031628 0.006047 0.014419 0.020814 0.044186 0.689535 0.199419**

**Min-Max Normalization (-1,1):**

**-0.985901 -0.981678 -1.000000 -0.994417 -0.996280 -0.954545 -0.973256 -0.804652 1.000000 -0.808559 -0.983789 -0.948837 -0.848837 -0.936744 -0.987906 -0.971161 -0.958372 -0.911628 0.379070 -0.601163**

**Z-Score Normalization:**

**-0.501749 -0.492246 -0.532957 -0.519953 -0.523455 -0.452634 -0.479446 -0.271702 2.442821 -0.277204 -0.496749 -0.446131 -0.325389 -0.425121 -0.505251 -0.475944 -0.459940 -0.385102 1.584636 0.247137**

**Decimal Scaled Normalization:**

**0.010000 0.012000 0.003000 0.006000 0.005000 0.025000 0.017000 0.100000 1.000000 0.098000 0.011000 0.027000 0.078000 0.033000 0.009000 0.018000 0.023000 0.044000 0.690000 0.200000**

### Practical 4

Implement Binning methods for data smoothing for the following dataset

using 3 equi-depth bins. Demonstrate (a) smoothing by bin means, (b)

smoothing by bin medians (c) bin boundaries.

Sample Dataset: 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70

#include <stdio.h>

#include <math.h>

#define SIZE 27

int data[SIZE] = {13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 5, 35, 35, 36, 40, 45, 46, 52, 70};

void print\_array(int array[], int size) {

int i;

for (i = 0; i < size; i++) {

printf("%d ", array[i]);

}

printf("\n");

}

void bin\_mean(int data[], int result[], int size, int bin\_depth) {

int i, j, sum, mean;

for (i = 0; i < size; i += bin\_depth) {

sum = 0;

for (j = i; j < i + bin\_depth && j < size; j++) {

sum += data[j];

}

mean = sum / bin\_depth;

for (j = i; j < i + bin\_depth && j < size; j++) {

result[j] = mean;

}

}

}

void bin\_median(int data[], int result[], int size, int bin\_depth) {

int i, j;

for (i = 0; i < size; i += bin\_depth) {

int median = data[i + bin\_depth / 2];

for (j = i; j < i + bin\_depth && j < size; j++) {

result[j] = median;

}

}

}

void bin\_boundaries(int data[], int result[], int size, int bin\_depth) {

int i, j, min, max;

for (i = 0; i < size; i += bin\_depth) {

min = data[i];

max = data[i + bin\_depth - 1 < size ? i + bin\_depth - 1 : size - 1];

for (j = i; j < i + bin\_depth && j < size; j++) {

if (fabs(data[j] - min) < fabs(data[j] - max)) {

result[j] = min;

} else {

result[j] = max;

}

}

}

}

int main() {

int result[SIZE];

int bin\_depth;

printf("Enter bin depth: ");

scanf("%d", &bin\_depth);

printf("Original Data:\n");

print\_array(data, SIZE);

bin\_mean(data, result, SIZE, bin\_depth);

printf("\nSmoothing by Bin Means:\n");

print\_array(result, SIZE);

bin\_median(data, result, SIZE, bin\_depth);

printf("\nSmoothing by Bin Medians:\n");

print\_array(result, SIZE);

bin\_boundaries(data, result, SIZE, bin\_depth);

printf("\nSmoothing by Bin Boundaries:\n");

print\_array(result, SIZE);

return 0;

}

**Output :**

Enter bin depth: 3

Original Data:

13 15 16 16 19 20 20 21 22 22 25 25 25 25 30 33 33 35 5 35 35 36 40 45 46 52 70

Smoothing by Bin Means:

14 14 14 18 18 18 20 20 20 24 24 24 25 25 25 33 33 33 25 25 25 40 40 40 47 47 47

Smoothing by Bin Medians:

15 15 15 19 19 19 21 21 21 25 25 25 25 25 25 33 33 33 35 35 35 45 45 45 46 46 46

Smoothing by Bin Boundaries:

13 13 16 16 19 20 20 20 22 22 25 25 25 25 30 30 33 35 5 35 35 36 40 40 45 52 70

### Practical 5

Generate a linear regression based model and demonstrate its working on

the given dataset.

#include <stdio.h>

#include <stdlib.h>

double simple\_linear\_reg(double \*x, double \*y, int n, int size) {

double sum\_x = 0, sum\_y = 0;

int i;

// Calculate sums of x and y

for (i = 0; i < size; i++) {

sum\_x += x[i];

sum\_y += y[i];

}

double mean\_x = sum\_x / size;

double mean\_y = sum\_y / size;

double m = 0, b = 0;

// Calculate slope (m)

for (i = 0; i < size; i++) {

m += (x[i] - mean\_x) \* (y[i] - mean\_y);

}

double denom = 0;

for (i = 0; i < size; i++) {

denom += (x[i] - mean\_x) \* (x[i] - mean\_x);

}

m /= denom;

// Calculate intercept (b)

b = mean\_y - (m \* mean\_x);

// Predict the price for the given area

double predicted\_price = (m \* n) + b;

return predicted\_price;

}

int main() {

double areas[] = {1500, 1600, 1700, 1800, 1900};

double prices[] = {300000, 320000, 340000, 360000, 380000};

int size = sizeof(areas) / sizeof(areas[0]);

int n;

printf("Enter the area of the house: ");

scanf("%d", &n);

double predicted\_price = simple\_linear\_reg(areas, prices, n, size);

printf("The predicted price for a house with area %d is: %.2f\n", n, predicted\_price);

return 0;

}

**Output :**

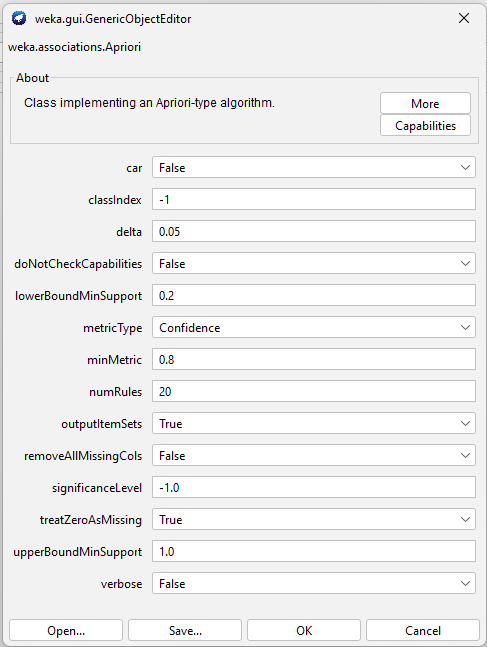
Enter the area of the house: 2000

The predicted price for a house with area 2000 is: 400000.00

### Practical 6

**Study and implement the Apriori algorithm over the given dataset. (Use WEKA**

**toolkit)**

****

**=== Run information ===**

**Scheme: weka.associations.Apriori -I -N 20 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0.2 -S -1.0 -Z -c -1**

**Relation: weather.symbolic**

**Instances: 14**

**Attributes: 5**

**outlook**

**temperature**

**humidity**

**windy**

**play**

**=== Associator model (full training set) ===**

**Apriori**

**=======**

**Minimum support: 0.2 (3 instances)**

**Minimum metric <confidence>: 0.8**

**Number of cycles performed: 16**

**Generated sets of large itemsets:**

**Size of set of large itemsets L(1): 7**

**Large Itemsets L(1):**

**outlook=overcast 4**

**outlook=rainy 5**

**temperature=mild 6**

**temperature=cool 4**

**humidity=normal 7**

**windy=FALSE 8**

**play=no 5**

**Size of set of large itemsets L(2): 6**

**Large Itemsets L(2):**

**outlook=rainy temperature=mild 3**

**outlook=rainy humidity=normal 3**

**outlook=rainy windy=FALSE 3**

**temperature=mild windy=FALSE 3**

**temperature=cool humidity=normal 4**

**humidity=normal windy=FALSE 4**

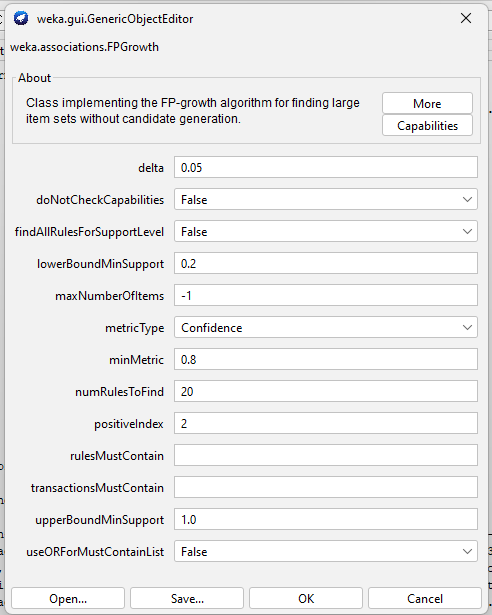
**Best rules found:**

**1. temperature=cool 4 ==> humidity=normal 4 <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)**

### Practical 7

Study and implement the FP-Growth algorithm over the given dataset. (Use

WEKA toolkit)



**=== Run information ===**

**Scheme: weka.associations.FPGrowth -P 2 -I -1 -N 20 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0.2**

**Relation: vote**

**Instances: 435**

**Attributes: 17**

**handicapped-infants**

**water-project-cost-sharing**

**adoption-of-the-budget-resolution**

**physician-fee-freeze**

**el-salvador-aid**

**religious-groups-in-schools**

**anti-satellite-test-ban**

**aid-to-nicaraguan-contras**

**mx-missile**

**immigration**

**synfuels-corporation-cutback**

**education-spending**

**superfund-right-to-sue**

**crime**

**duty-free-exports**

**export-administration-act-south-africa**

**Class**

**=== Associator model (full training set) ===**

**FPGrowth found 20 rules (displaying top 20)**

**1. [adoption-of-the-budget-resolution=y, anti-satellite-test-ban=y]: 201 ==> [aid-to-nicaraguan-contras=y]: 187 <conf:(0.93)> lift:(1.67) lev:(0.17) conv:(5.95)**

**2. [el-salvador-aid=y]: 212 ==> [religious-groups-in-schools=y]: 197 <conf:(0.93)> lift:(1.49) lev:(0.15) conv:(4.96)**

**3. [crime=y, el-salvador-aid=y]: 194 ==> [religious-groups-in-schools=y]: 180 <conf:(0.93)> lift:(1.48) lev:(0.13) conv:(4.85)**

**4. [mx-missile=y]: 207 ==> [aid-to-nicaraguan-contras=y]: 192 <conf:(0.93)> lift:(1.67) lev:(0.18) conv:(5.74)**

**5. [el-salvador-aid=y]: 212 ==> [crime=y]: 194 <conf:(0.92)> lift:(1.61) lev:(0.17) conv:(4.8)**

**6. [religious-groups-in-schools=y, el-salvador-aid=y]: 197 ==> [crime=y]: 180 <conf:(0.91)> lift:(1.6) lev:(0.16) conv:(4.7)**

**7. [aid-to-nicaraguan-contras=y, anti-satellite-test-ban=y]: 210 ==> [adoption-of-the-budget-resolution=y]: 187 <conf:(0.89)> lift:(1.53) lev:(0.15) conv:(3.66)**

**8. [superfund-right-to-sue=y]: 209 ==> [religious-groups-in-schools=y]: 186 <conf:(0.89)> lift:(1.42) lev:(0.13) conv:(3.26)**

**9. [aid-to-nicaraguan-contras=y]: 242 ==> [adoption-of-the-budget-resolution=y]: 215 <conf:(0.89)> lift:(1.53) lev:(0.17) conv:(3.62)**

**10. [mx-missile=y]: 207 ==> [anti-satellite-test-ban=y]: 182 <conf:(0.88)> lift:(1.6) lev:(0.16) conv:(3.59)**

**11. [anti-satellite-test-ban=y]: 239 ==> [aid-to-nicaraguan-contras=y]: 210 <conf:(0.88)> lift:(1.58) lev:(0.18) conv:(3.53)**

**12. [adoption-of-the-budget-resolution=y, aid-to-nicaraguan-contras=y]: 215 ==> [anti-satellite-test-ban=y]: 187 <conf:(0.87)> lift:(1.58) lev:(0.16) conv:(3.34)**

**13. [mx-missile=y]: 207 ==> [adoption-of-the-budget-resolution=y]: 180 <conf:(0.87)> lift:(1.5) lev:(0.14) conv:(3.09)**

**14. [aid-to-nicaraguan-contras=y]: 242 ==> [anti-satellite-test-ban=y]: 210 <conf:(0.87)> lift:(1.58) lev:(0.18) conv:(3.3)**

**15. [crime=y]: 248 ==> [religious-groups-in-schools=y]: 214 <conf:(0.86)> lift:(1.38) lev:(0.14) conv:(2.66)**

**16. [adoption-of-the-budget-resolution=y]: 253 ==> [aid-to-nicaraguan-contras=y]: 215 <conf:(0.85)> lift:(1.53) lev:(0.17) conv:(2.88)**

**17. [el-salvador-aid=y]: 212 ==> [religious-groups-in-schools=y, crime=y]: 180 <conf:(0.85)> lift:(1.73) lev:(0.17) conv:(3.26)**

**18. [superfund-right-to-sue=y]: 209 ==> [crime=y]: 177 <conf:(0.85)> lift:(1.49) lev:(0.13) conv:(2.72)**

**19. [religious-groups-in-schools=y, crime=y]: 214 ==> [el-salvador-aid=y]: 180 <conf:(0.84)> lift:(1.73) lev:(0.17) conv:(3.13)**

**20. [anti-satellite-test-ban=y]: 239 ==> [adoption-of-the-budget-resolution=y]: 201 <conf:(0.84)> lift:(1.45) lev:(0.14) conv:(2.56)**