v.

* Assumptions:

1. The cuisine types are divided manually and are most of the origination of the food type.
2. The restaurants that do not have vegetarian (243) feature are considered not offering vegetarian.
3. The first cuisine type encountered in the restaurant data is extracted for association rule mining.
4. Weka is used to mine the association rules between the three attributes: cuisine, price and vegietarian.

* Results:

The Tertius method works well to mine the association rules for vegetarian restaurant. Below please find the results from Weka. The default scheme was used (weka.associations.Tertius -K 10 -F 0.0 -N 1.0 -L 4 -G 0 -c 0 -I 0 -P 0).

Tertius

=======

1. /\* 0.098122 0.017548 \*/ price = 162 ==> vegie = Yes or cuisine = 221

2. /\* 0.094522 0.016346 \*/ price = 162 ==> vegie = Yes or cuisine = 229

3. /\* 0.093922 0.018029 \*/ price = 162 ==> vegie = Yes or cuisine = 058

4. /\* 0.093168 0.018510 \*/ price = 162 ==> cuisine = 221

5. /\* 0.092312 0.017548 \*/ price = 162 ==> vegie = Yes or cuisine = 142

6. /\* 0.091351 0.018269 \*/ price = 162 ==> vegie = Yes or cuisine = 009

7. /\* 0.089326 0.017308 \*/ price = 162 ==> cuisine = 229

8. /\* 0.088972 0.018990 \*/ price = 162 ==> cuisine = 058

9. /\* 0.088802 0.018269 \*/ price = 162 ==> cuisine = 142

10. /\* 0.088764 0.018990 \*/ price = 162 ==> vegie = Yes or cuisine = 186

Number of hypotheses considered: 10543

Number of hypotheses explored: 6667

* Perl code to generate the .arff file:

use strict;

use warnings;

my @cuisine = split /,/, "002,003,005,006,007,008,009,012,013,014,015,018,020,022,031,032,033,034,038,039,045,048,049,058,067,069,070,072,082,088,089,090,091,092,093,094,095,096,098,103,105,110,116,117,118,119,120,121,122,123,124,125,126,127,128,129,130,131,133,134,135,139,140,142,143,144,147,171,172,179,180,186,187,188,195,199,201,202,209,210,211,218,219,220,221,222,223,227,228,229,235,236,238,239,240,241,244,246,256";

my %cuisine\_elements;

@cuisine\_elements{@cuisine} = ();

my @price = split /,/, "161,162,163,164,165,166,167,168,169,170";

my %price\_elements;

@price\_elements{@price} = ();

my $dir = 'C:/Users/yhan/Documents/data';

my $out = 'C:/Users/yhan/Documents/CS773/q5/vegie\_cuisinePrice.arff';

open(my $file, '>', $out) or die "Could not open file '$out' $!";

foreach my $fp (glob("$dir/\*.txt")) {

open my $fh, "<", $fp or die "can't read open '$fp'";

while (<$fh>) {

chomp;

my $line = $\_;

my @features = split /[\s,\t]/, $\_;

my @rescui = grep exists $cuisine\_elements{$\_}, @features;

if (length($rescui[0]) == 0) {$rescui[0] = '?'}

my @respri = grep exists $price\_elements{$\_}, @features;

if (length($respri[0]) == 0) {$respri[0] = '?'}

if ($line =~ /243/)

{

printf $file "$rescui[0], $respri[0], Yes \n"

}

else {

printf $file "$rescui[0], $respri[0], No \n"

};

}

close $fh or die "can't read close";

}