Course Project

CS773: Data Mining and Security

Summer 2016

John Berlin, Yun Han

Contents

List of Tables

List of Figures

Listings

Data Changes

For questions two through four the data was reduced to *cuisine,atmosphere,occasion,price,style*. Any restaurant that did not have one of those features the fake feature none was added and for those that had multiple features per feature category i.e one atmosphere, cuisine and price but two occasion features with non style the following was done. This is shown in table 1.

Data change example

cuisine	atmosphere	occasion	price	style
Italian	Excellent Decor	Open on Sundays	$15-$30	none
Italian	Excellent Decor	Open on Mondays	$15-$30	none

Task i

Study the provided features and classify them into one of the standard (cuisine, style, price, atmosphere, and occasion) or into your own created additional categories. Limit the new categories to at most 5. If you think that a feature fits into more than one of your categories, do put them in all the categories that they fit in. This could be more an exception than a role. Typically, there should be a 1-1 mapping of features to categories.

Results

Atmosphere Classification

An Historic Spot	An Out Of The Way Find	Authentic
Buffet Dining	Business Scene	Cafe/Garden Dining
Classic Hotel Dining	Creative	Credit cards are not accepted
Excellent Decor	Excellent Food	Excellent Service
Extraordinary Decor	Extraordinary Food	Extraordinary Service
Fabulous Views	Fabulous Wine Lists	Fair Decor
Fair Food	Fair Service	Focus on Dessert
For the Young and Young at Heart	Good Decor	Good Food
Good Out of Town Business	Good Service	Good for Younger Kids
Great for People Watching	Health Conscious Menus	Hip Place To Be
Little Known But Well Liked	Near-perfect Decor	Near-perfect Food
Near-perfect Service	Need To Dress	No Liquor Served
No Reservations	No Smoking Allowed	Old World Cafe Charm
On the Beach	Parking/Valet	People Keep Coming Back
Place for Singles	Poor Decor	Pub Feel
Quiet for Conversation	Quirky	Relaxed Senior Scene
Romantic	Singles Scene	Tourist Appeal
Up and Coming	Very Busy - Reservations a Must	Warm spots by the fire
Wheelchair Access		

Cuisine Classification

Afghanistan
African
American
Argentinean
Armenian
Asian
Austrian
Bar-B-Q
Belgian
Brazilian
Burmese
Burritos
Cajun
Cambodian
Canadian
Caribbean
Chinese
Coffee and Dessert
Creole
Cuban
Czech
Dim Sum
Egyptian
Ethiopian
Filipino
Fountain and Ice Cream
French
German
Greek
Guatemalan
Hamburgers
Beer & Hot Dogs
Hungarian
Indian
Indonesian
Irish
Italian
Jamaican
Japanese
Jewish
Korean
Latin
Lebanese
Malaysian
Mediterranean
Mexican
Moroccan
Nicaraguan
Pacific New Wave
Pacific Rim
Persian
Peruvian
Polish
Polynesian
Portuguese
Puerto Rican
Romanian
Roumanian
Russian
Salvadoran
Scandinavian
Seafood
Spanish
Sushi
Swiss
Tapas
Tex Mex
Tex-Mex
Thai
Tibetan
Tunisian
Turkish
Ukrainian
Ukranian
Vegetarian
Venezuelan
Vietnamese
Wine and Beer
Yugoslavian

Occasion Classification

After Hours Dining
Catering for Special Events
Dancing
Delivery Available
Dining After the Theater
Dining Outdoors
Early Dining
Entertainment
Fine for Dining Alone
Game
Great Place to Meet for a Drink
Happy Hour
Late Night Menu
Long Drive
Margaritas
Menus in Braille
Open for Breakfast
Open on Mondays
Open on Sundays
Other Quick Food
Parties and Occasions
Picnics
Pre-theater Dining
Private Parties
Private Rooms Available
Prix Fixe Menus
See the Game
Short Drive
Special Brunch Menu
Takeout Available
Walk
Weekend Brunch
Weekend Dining
Weekend Jazz Brunch
Weekend Lunch

Style Classification

A
American (Contemporary)
American (New)
American (Regional)
American (Traditional)
Bakeries
Brasserie
Cab
Cafe/Espresso Bars
Cafeterias
Californian
Carry in Wine and Beer
Caviar
Central
Coffee Houses
Continental
Haute Creole
Haute New Orleans
Coffeehouses
Coffee Shops
Deli
Diners
Down-Home
Eastern European
Eclectic
Down-Home Creole
Southern
Southwestern
South American
Southeast Asian
English
Fast Food
Fondue
Franco-Russian
Frankfurters
French Bistro
French Classic
French Contemporary
French-Japanese
French (New)
French Nouvelle
Grills
Hamburgers
Health Food
High Tea
International
Italian (North
Italian (Northern)
Italian (North & South)
Italian Nuova Cucina
Italian (Southern)
Kosher
Lithuanian
Middle Eastern
N
Noodle Houses
Noodle Shops
Omelettes
Oyster Bars
Pancakes
Pastries
Pastry Shops
Pizza
Pizzerias
Po' Boys
Power Brokers
Scottish
Soul Food
Soulfood
Southern Comfort
Steakhouses
Swiss-French
Tacos
Traditional
Yogurt Bar

Price Classification

$15-$30 $30-$50 below $15 over $50

Task ii

Analyze and form rules to characterize the following five types of cuisine: (i) Indian (ii) Mexican (iii) Italian (iv) French (v) American. For each category, derive a set of rules based on data available from all cities.

Results

The results for task ii were generated from an R script seen in listing 3 which uses the C50 decision tree algorithm to produce rules. Data used in this script was generated using a python file `rules.py` which accompanies this report. The classification was aided through using boosting which is indicated by setting the *trialNum* to ten as well as setting predictor winnowing(feature selection) to true. The ruleGen file can be seen in this listing 4 in the appendices for this report. We also set the flag of rules to true in order to decompose the tree into the bare rules.

Overall the classification of the data the features *price*, *style*, *occasion* were heavily used when generating the rules as seen in table 9. Boosting resulted in a `38.7%` error margin with trials 0, 7 and 8 resulting in least error. The entire rule set for each trial can be found in the files *c50_10Trial(2).txt* which accompanies this report. Trial 0 generated the most number of rules which was 118 with the average number of rules being between 40-60. We omit listing the rules here as the file is 5977 lines long but include notable rules as seen below. This is a sampling seen in figure 1 is taken from all trials.

```
Rule 0/24: (582, lift 6.4)
style in {Bakeries, Brasserie, Caviar}
->  class French  [0.998]

Rule 0/38: (3, lift 5.1)
atmosphere = Romantic
occasion = After Hours Dining
price = $30-$50
style = none
->  class French  [0.800]

Rule 0/58: (4, lift 56.3)
atmosphere in {Good Food, Good Service}
price = $15-$30
style = Carry in Wine and Beer
->  class Indian  [0.833]

Rule 0/106: (8, lift 11.5)
occasion in {Dining Outdoors, Late Night Menu}
price = below $15
style = Carry in Wine and Beer
->  class Mexican  [0.900]

Rule 3/4: (1984.1/437.8, lift 1.8)
occasion in {Dancing, Delivery Available, Dining After the Theater,
                     Early Dining, Fine for Dining Alone,
                     Great Place to Meet for a Drink, Long Drive,
                     Menus in Braille, Open on Mondays, Open on Sundays,
                     Picnics, See the Game, Short Drive, Special Brunch Menu,
                     Takeout Available, Weekend Dining}
style in {Coffee Shops, Eclectic, High Tea, Omelettes, Pastries,
                  Steakhouses}
->  class American  [0.779]
```

Sampling of rules from C50

Task ii Classification Errors

```
Evaluation on training data (61063 cases):
Trial         Rules
-----  ----------------
    No      Errors

   0    119 22584(37.0%)
   1     60 25297(41.4%)
   2     61 24058(39.4%)
   3     45 25904(42.4%)
   4     61 24293(39.8%)
   5     53 25591(41.9%)
   6     43 25873(42.4%)
   7     78 23947(39.2%)
   8     65 23669(38.8%)
   9     52 24269(39.7%)
boost       23614(38.7%)
```

Task ii Classification Confusion Matrix

```
 (a)   (b)   (c)   (d)   (e)    <-classified as
 ----  ----  ----  ----  ----
25111  1206        2420   360    (a): class American
 4754  3533        1220    10    (b): class French
  640    15         229    20    (c): class Indian
 8781   572        7247   150    (d): class Italian
 2769               468  1558    (e): class Mexican
```

Task ii Attribute Usage

```
100.00% price
100.00% style
 99.99% occasion
 89.66% atmosphere
```

Task iii

Let us now concentrate specifically on the quality of the food. This is specified through features 73-78. Assuming that the outcome you are interested is in one of the following categories, determine if there is any relationship between type of cuisine and the quality indicator. Categories to be considered are: Fair, Good, Excellent. You can combine the given 6 categories into these 3 categories.

Results

Task iv

Derive association rules among the given features. In other words, does Creative atmosphere imply a specific category? In particular, experiment with the following associations:
a. Cuisine and atmosphere
b. Price and atmosphere
c. Price and style
d. Cuisine and occasion.
e. Dcor and Price

Results

Task v

Find an association between a restaurant offering vegetarian (243) to its price and cuisine.

Results

1. Assumptions

   (a) The cuisine types are divided manually and are most of the origination of the food type.

   (b) The restaurants that do not have vegetarian (243) feature are considered not offering vegetarian

   (c) The first cuisine type encountered in the restaurant data is extracted for association rule mining.

   (d) Weka is used to mine the association rules between the three attributes: cuisine, price and vegietarian

The Tertius method works well to mine the association rules for vegetarian restaurant. Below in figure 2 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). The perl code used for the task is seen in listing 1.

```
 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
```

Tertius results

Task vi

Determine the error that would be incurred by categorizing the restaurants based on the continents they represent: Asia, Europe, Africa, North America, and South America. For each continent, form rules to determine the outcome (which continent they come from) based on other attributes such as price, atmosphere, quality of food, etc.

Results

1. Assumptions

   (a) The restaurants that contain geographical information features can be accurately categorized by continent.

   (b) The list of countries and continents is a complete list.

   (c) The restaurants that do not match the list of countries and continents cannot be categorized by continents and generate errors.

The total number of restaurants: 4160. The number of restaurants that contain geographical features: 3758.

$$Error = \frac{4160 - 3758}{4160} * 100\% = 9.66\%$$

Perl code to get the number of total restaurants and the number of restaurants that contain geographical features can bee seen in listing 2.

Appendices

Perl Scripts

```
use strict;
use warnings;

my @cuisine = split /,/, "
    002,003,005,006,007,008,009,012,013,014,015,018,020,022,031,032,033,034,038,039,045,048,049,058,067,069,070,07
    ";
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";
}
```

Task v Perl script

```
use strict;
use warnings;

my $features = 'C:/Users/yhan/Documents/CS773/q6/features.txt';
open(my $featurefile, '<', $features) or die "Could not open file!";
my %feature;
while (<$featurefile>) {
  my $line = $_;
  my ($i, $j) = split(/\t/, $line);
  $feature{$i} = $j;
  #print "$i is $feature{$i}\n";
  }


my @countries;
my $country = 'C:/Users/yhan/Documents/CS773/q6/countries.txt';
open(my $file, '<', $country) or die "Could not open file!";
while(<$file>) {
  chomp;
  my $line = $_;
  push(@countries, $line);
  #print $_;
  }
#my %country_elements;
#@country_elements{@countries} = ();

my $countrys = "African Afro-Eurasian ...";
my $total = 0;
my $geo = 0;

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

```perl
foreach my $fp (glob("$dir/*.txt")) {
  open my $fh, "<", $fp or die "can't read open '$fp'";
  while (<$fh>) {
    chomp;
    my $line = $_;
    $total = $total + 1;
    my $flag = 0;
    my @resfeatures = split /[\s,\t]/, $_;
    foreach my $ele (@resfeatures)
    {
    if(exists $feature{$ele}){
      my $cty = $feature{$ele};
      $cty =~ tr/A-Za-z//cd;
      if ($countrys =~ /$cty/)
      {
      $flag = 1
      #print $feature{$ele};
      }

    #print $feature{$ele};
    }
          }
    $geo = $geo + $flag;
    }
close $fh or die "can't read close";
}

print "$total, $geo \n";
```

Task vi Perl script

R Scripts

```r
#!/usr/bin/Rscript
cwd <- getwd()
setwd(cwd)

#sink to redirect stdout to file

source(file=file.path(cwd,'/r/ruleFunctions.R'))

s1 <- read.csv('q2/cuisineCharacters2.csv')

ruleModel <- ruleGen.c50(d=s1,form =  cuisine ~ .,trialNum = 10,winnow = TRUE)


c50Summary <- capture.output(summary(ruleModel))
cat(c50Summary, file = file.path(cwd,'q2','c50_10Trials2.txt'),sep="\n")
```

Task ii R script

```r
library(caret)
library(arules)
library(C50)
library(Rsenal)
library(rpart)


ruleGen.c50 <- function(d,
                        form,
                        trialNum = 4,
                        winnow = FALSE) {
  C5.0(
    formula = form,
    data = d,
    trials = trialNum,
    control = C5.0Control(subset = TRUE, winnow = winnow),
    rules = TRUE
  )
}

ruleGen.apriori <-
```

```r
  function(data, minLen = 2, maxLen = 50,
           sup = 0.005, conf = 0.005, targ = 'rules',
           rH = NULL, lH = NULL, deflt = 'lhs',list=F) {
    appear <- NULL
    lhNull <- is.null(lH)
    rhNull <- is.null(rH)
    if (!lhNull && !rhNull) {
      appear <- list(rhs = rH,
                     lhs = lH,
                     default = deflt)
    } else if (!lhNull && rhNull) {
      appear <- list(lhs = lH,
                     default = deflt)
    } else if (lhNull && !rhNull) {
      appear <- list(rhs = rH,
                     default = deflt)
    } else {
      appear <- list(default = 'both')
    }

    rules <- apriori(
      data,
      parameter = list(
        minlen = minLen,
        maxlen = maxLen,
        supp = sup,
        confidence = conf,
        target = targ
      ),
      appearance = appear,
      control = list(verbose = F, filter = 0)
    )

    if(length(rules) > 0) {
      rules2df(addRuleQuality(
        trans = data,
        rules = rules,
        exclude =  c(
          "hyperConfidence",
          "cosine",
          "chiSquare",
          "coverage",
          "doc",
          "gini",
          "hyperLift",
          "fishersExactTest",
          "improvement",
          "leverage",
          "oddsRatio",
          "phi",
          "RLD"
        )
      ), list = list)
    } else {
      'no rules'
    }

}

ruleGen.part <- function(data, form) {
  train(form, data = data, method = "PART")
}

ruleGen.appearance_pair_list <- function(df,notFromHelper=F) {
  # browser()
  if(notFromHelper){
    unlist(c(levels(unique(df$s1)), levels(unique(df$s2))),use.names=F)
  } else {
    unlist(c(unique(df$s1), unique(df$s2)),use.names=F)
    # isS1_list = is.list(df$s1)
    # isS2_list = is.list(df$s2)
    #
    # if(isS1_list && isS2_list) {
    #   unlist(c(unique(df$s1), unique(df$s2)),use.names=F)
    # } else if(!isS1_list && isS2_list){
    #   unlist(c(df$s1, unique(df$s2)),use.names=F)
    # } else {
    #   unlist(c(unique(df$s1), df$s2),use.names=F)
    # }
  }
}

ruleGen.rules_apriori_lh <- function(data,feature_pair,notFromHelper=F,sup=0.0005, conf = 0.0005,minLen = 2, maxLen = 50, list=F)  {
  rlh <- ruleGen.appearance_pair_list(feature_pair, notFromHelper = notFromHelper)
  ruleGen.apriori(
    data,
    lH = rlh,
    sup = sup,
    conf = conf,
    minLen = minLen,
    maxLen = maxLen,
    deflt = 'rhs',
    list=list
  )
}

ruleGen.rules_apriori_lhGrouped <- function(data,lhs_grouped,notFromHelper=F,sup=0.0005, conf = 0.0005, minLen = 2, maxLen = 50, list=F)  {

  ruleGen.apriori(
    data,
    lH = rlh,
    sup = sup,
    conf = conf,
    minLen = minLen,
    maxLen = maxLen,
    deflt = 'rhs',
    list=list
  )
}

ruleGen.rules_to_df <- function(rules,list=F){
  rules2df(rules,list=list)
}
```

Rule functions

```
library(dplyr)
library(tidyr)
library(purrr)
library(stringr)
library(reshape2)
library(foreach)
library(iterators)

setwd(getwd())

helpers.filter_df <- function(data,...) {
  data %>% filter(...)
}

helpers.unqiue_all <- function(data) {

  ucuisine <- data %>% select(cuisine) %>% distinct
  uatmosphere <- data %>% select(atmosphere) %>% distinct
  uoccasion <- data %>% select(occasion) %>% distinct
  uprice <- data %>% select(price) %>% distinct
  ustyle <- data %>% select(style) %>% distinct

  list(
   c = ucuisine,
   a = uatmosphere,
   o = uoccasion,
   p = uprice,
   s = ustyle
  )
}

helpers.test <- function(d,sides,...) {
  # it <- sides %>% by_row(function(r)
```

```r
  #   c(str_trim(r$s1, side = c(
  #     "both", "left", "right"
  #   )), str_trim(r$s2, side = c(
  #     "both", "left", "right"
  #   ))), .to = 'rdf')
  #
   foreach(r = iter(sides,by='row'),.combine = c) %do% {
    print(d %>% select(cuisine == r$s1 & atmosphere == r$s2))
    pair <- c(r$s1,r$s2)
    1
  }
  # View(it)
}




helpers.s1_to_groupedS2 <- function(d,sides,rulesFun) {
  sides %>% group_by(s1) %>% summarise(s2 = list(s2)) %>%
    by_row(function(r) unlist(list(r$s1,r$s2)), .to='lhs') %>%
    by_row(function(r)  do.call(rulesFun,list(data=d,lH = r$lhs,  deflt = 'rhs' )), .to='rdf')
}


helpers.s1_s2_pairs <- function(sides) {
  sides %>% mutate(lhs1 = paste(s1,s2,sep=','),lhs2 = paste(s2,s1,sep=','))
}


helpers.pair_count <- function(df) {
  df %>% summarise()
}

helpers.q3 <- function(write = F, filter = NULL) {
  data <-
    read.csv(file.path('q3','restaurantsq3.csv'))

  filter <- !is.null(filter)

  cuisine_atmosphere <-
    data %>% select(cuisine, atmosphere) %>% distinct %>%
    filter(cuisine != 'none' & atmosphere != 'none') %>%
    transmute(
      s1 = paste('cuisine', cuisine, sep = '='),
      s2 = paste('atmosphere', atmosphere, sep = '=')
    )

  cuisine_occasion <-
    data %>% select(cuisine, occasion) %>% distinct %>%
    filter(cuisine != 'none' & occasion != 'none') %>%
    transmute(
      s1 = paste('cuisine', cuisine, sep = '='),
      s2 = paste('occasion', occasion, sep = '=')
    )

  cuisine_price <-
    data %>% select(cuisine, price) %>% distinct  %>%
    filter(cuisine != 'none' & price != 'none') %>%
    transmute(s1 = paste('cuisine', cuisine, sep = '='),
              s2 = paste('price', price, sep = '='))

  cuisine_style <-
    data %>% select(cuisine, style) %>% distinct  %>%
    filter(cuisine != 'none' & style != 'none') %>%
    transmute(s1 = paste('cuisine', cuisine, sep = '='),
              s2 = paste('style', style, sep = '='))

  atmosphere_occasion <-
    data %>% select(atmosphere, occasion) %>% distinct  %>%
    filter(atmosphere != 'none' & occasion != 'none') %>%
    transmute(
      s1 = paste('atmosphere', atmosphere, sep = '='),
      s2 = paste('occasion', occasion, sep = '=')
    )

  atmosphere_price <-
```

```r
    data %>% select(atmosphere, price) %>% distinct  %>%
    filter(atmosphere != 'none' & price != 'none') %>%
    transmute(
      s1 = paste('atmosphere', atmosphere, sep = '='),
      s2 = paste('price', price, sep = '=')
    )

  atmosphere_style <-
    data %>% select(atmosphere, style) %>% distinct  %>%
    filter(atmosphere != 'none' & style != 'none') %>%
    transmute(
      s1 = paste('atmosphere', atmosphere, sep = '='),
      s2 = paste('style', style, sep = '=')
    )

  occasion_price <-
    data %>% select(occasion, price) %>% distinct  %>%
    filter(occasion != 'none' & price != 'none') %>%
    transmute(
      s1 = paste('occasion', occasion, sep = '='),
      s2 = paste('price', price, sep = '=')
    )

  occasion_style <-
    data %>% select(occasion, style) %>% distinct  %>%
    filter(occasion != 'none' & style != 'none') %>%
    transmute(
      s1 = paste('occasion', occasion, sep = '='),
      s2 = paste('style', style, sep = '=')
    )

  style_price <-
    data %>% select(style, price) %>% distinct  %>%
    filter(style != 'none' & price != 'none') %>%
    transmute(s1 = paste('style', style, sep = '='),
              s2 = paste('price', price, sep = '='))

  decor_price <-
    data %>% select(atmosphere, price) %>% distinct  %>%
    filter(atmosphere != 'none' &
             price != 'none' & grepl('Decor', atmosphere)) %>%
    transmute(
      s1 = paste('atmosphere', atmosphere, sep = '='),
      s2 = paste('price', price, sep = '=')
    )

  if (write) {
    dataForR <- file.path('..', 'r_data')
    dir.create(dataForR, showWarnings = F)

    write.csv(
      cuisine_atmosphere,
      file = file.path(dataForR, 'q3_cuisine_atmosphere.csv'),
      fileEncoding = 'utf8',
      row.names = F
    )

    write.csv(
      cuisine_occasion,
      file = file.path(dataForR, 'q3_cuisine_occasion.csv'),
      fileEncoding = 'utf8',
      row.names = F
    )

    write.csv(
      cuisine_price,
      file = file.path(dataForR, 'q3_cuisine_price.csv'),
      fileEncoding = 'utf8',
      row.names = F
    )

    write.csv(
      cuisine_style,
      file = file.path(dataForR, 'q3_cuisine_style.csv'),
      fileEncoding = 'utf8',
      row.names = F
    )

    write.csv(
      atmosphere_occasion,
      file = file.path(dataForR, 'q3_atmosphere_occasion.csv'),
      fileEncoding = 'utf8',
      row.names = F
    )

    write.csv(
      atmosphere_price,
      file = file.path(dataForR, 'q3_atmosphere_price.csv'),
      fileEncoding = 'utf8',
      row.names = F
    )

    write.csv(
      atmosphere_style,
      file = file.path(dataForR, 'q3_atmosphere_style.csv'),
      fileEncoding = 'utf8',
      row.names = F
    )

    write.csv(
      occasion_price,
      file = file.path(dataForR, 'q3_occasion_price.csv'),
      fileEncoding = 'utf8',
      row.names = F
    )

    write.csv(
      occasion_style,
      file = file.path(dataForR, 'q3_occasion_style.csv'),
      fileEncoding = 'utf8',
      row.names = F
    )

    write.csv(
      style_price,
      file = file.path(dataForR, 'q3_style_price.csv'),
      fileEncoding = 'utf8',
      row.names = F
    )

    write.csv(
      decor_price,
      file = file.path(dataForR, 'q3_decor_price.csv'),
      fileEncoding = 'utf8',
      row.names = F
    )
  }

  list(
    q3Data = data,
    ca = cuisine_atmosphere,
    co = cuisine_occasion,
    cp = cuisine_price,
    cs = cuisine_style,
    ao = atmosphere_occasion,
    ap = atmosphere_price,
    as = atmosphere_style,
    op = occasion_price,
    os = occasion_style,
    sp = style_price,
    dp = decor_price
  )
}



helpers.q4 <- function(write = F) {
  data <- read.csv(file.path('q3', 'restaurantsq3.csv'))

  cuisine_foodQuality <-
    data %>% select(cuisine, atmosphere) %>%
    filter(grepl('Food', atmosphere))

  cuisine_foodQualitySide <-
    cuisine_foodQuality %>% distinct %>%
    transmute(
      s1 = paste('cuisine', cuisine, sep = '='),
      s2 = paste('atmosphere', atmosphere, sep = '=')
    )

  cuisine_foodQuality <- cuisine_foodQuality %>% transmute(cuisine=cuisine ,quality = atmosphere)

  cuisine_foodQuality <- droplevels(cuisine_foodQuality)
  cq_spread <- cuisine_foodQuality %>% group_by(cuisine,quality) %>% tally %>% spread(quality,n,fill = 0)

  cuisine_foodQualityD = cuisine_foodQuality
  cuisine_foodQualityD$quality <-
    recode_factor(
      cuisine_foodQualityD$quality,
      ‘Excellent Food‘ = ”Good”,
      ‘Near-perfect Food‘ = ”Excellent”,
      ‘Extraordinary Food‘ = ”Excellent”,
      ‘Fair Food‘ = 'Fair',
      ‘Good Food‘ = 'Good'
    )

  cqD_spread <- cuisine_foodQualityD %>% group_by(cuisine,quality) %>% tally %>% spread(quality,n,fill = 0)


  # cuisine_foodQuality$cuisine <- levels(cuisine_foodQuality$cuisine)
  # cuisine_foodQuality$quality <- levels(cuisine_foodQuality$quality)

  list(q4Data = data,
       cfqSide = cuisine_foodQualitySide,
       cfq = cuisine_foodQuality,
       cfqd = cuisine_foodQualityD,
       cqD_spread = cqD_spread,
       cq_spread=cq_spread)
}


helpers.pair_count <- function(df) {
  ldf <- df %>% transmute(
    cuisine = levels(unique(cuisine)),
    atmosphere = levels(unique(atmosphere)),
    occasion = levels(unique(occasion)),
    price = levels(unique(price)),
    style = levels(unique(style))
  )

  list(
      cuisine_atmosphere = df %>% select(cuisine, atmosphere) %>% group_by(cuisine, atmosphere) %>% tally(sort = TRUE),
      cuisine_occasion = df %>% select(cuisine, occasion) %>% group_by(cuisine, occasion) %>% tally(sort = TRUE),
      cuisine_price = df %>% select(cuisine, price) %>% group_by(cuisine, price) %>% tally(sort = TRUE),
      cuisine_style = df %>% select(cuisine, atmosphere) %>%  group_by(cuisine, atmosphere) %>% tally(sort = TRUE),
      atmosphere_occasion = df %>% select(cuisine, style) %>% group_by(cuisine, style) %>% tally(sort = TRUE),
      atmosphere_price = df %>% select(atmosphere, price) %>% group_by(atmosphere, price) %>% tally(sort = TRUE),
      atmosphere_style = df %>% select(atmosphere, style) %>% group_by(atmosphere, style) %>% tally(sort = TRUE),
      occasion_price = df %>% select(occasion, price) %>% group_by(occasion, price) %>% tally(sort = TRUE),
      occasion_style = df %>% select(occasion, style) %>% group_by(occasion, style) %>% tally(sort = TRUE),
      style_price = df %>% select(style, price) %>% group_by(style, price) %>% tally(sort = TRUE)
  )
}

helpers.q2_appearance <- function() {
  cuisine_appearance <-
    data %>% distinct(cuisine) %>% transmute(side = paste('cuisine', cuisine, sep = '='))

  atmosphere_appearance <-
```

```
    data %>% distinct(atmosphere) %>% transmute(side = paste('atmosphere', atmosphere, sep = '='))

  occasion_appearance <-
    data %>% distinct(occasion) %>% transmute(side = paste('occasion', occasion, sep = '='))

  price_appearance <-
    data %>% distinct(price) %>% transmute(side = paste('price', price, sep = '='))

  style_appearance <-
    data %>% distinct(style) %>% transmute(side = paste('style', style, sep = '='))

  write.csv(
    cuisine_appearance,
    file = file.path(dataForR, 'q2_cuisine_ap.csv'),
    fileEncoding = 'utf8'
  )
  write.csv(
    atmosphere_appearance,
    file = file.path(dataForR, 'q2_atmosphere_ap.csv'),
    fileEncoding = 'utf8'
  )
  write.csv(
    occasion_appearance,
    file = file.path(dataForR, 'q2_occasion_ap.csv'),
    fileEncoding = 'utf8'
  )
  write.csv(
    price_appearance,
    file = file.path(dataForR, 'q2_price_ap.csv'),
    fileEncoding = 'utf8'
  )
  write.csv(
    style_appearance,
    file = file.path(dataForR, 'q2_style_ap.csv'),
    fileEncoding = 'utf8'
  )
}

helpers.group_by_lhs <- function(ruleDF) {
  ruleDF %>% select(lhs, rhs, support, confidence, lift, conviction) %>% group_by(lhs) %>% summarise(
    associated = list(rhs),
    cumSup = sum(support),
    cumConf = sum(confidence),
    cumLift = sum(lift),
    sumConv = sum(conviction)
  )
}

# from http://www.r-bloggers.com/measuring-associations-between-non-numeric-variables/
helpers.GKtau <- function(x,y){
  #
  #  First, compute the IxJ contingency table between x and y
  #
  Nij <- table(x,y,useNA='ifany')
  #
  #  Next, convert this table into a joint probability estimate
  #
  PIij <- Nij/sum(Nij)
  #
  #  Compute the marginal probability estimates
  #
  PIiPlus = apply(PIij,MARGIN=1,sum)
  PIPlusj = apply(PIij,MARGIN=2,sum)
  #
  #  Compute the marginal variation of y
  #
  Vy <- 1 - sum(PIPlusj^2)
  #
  #  Compute the expected conditional variation of y given x
  #
  InnerSum <- apply(PIij^2,MARGIN=1,sum)
  VyBarx <-  1 - sum(InnerSum/PIiPlus)
  #
  #  Compute and return Goodman and Kruskal s tau measure
  #
  tau <- (Vy - VyBarx)/Vy
  tau
}
```

Helper functions