M6L1 Homework Assignment

Joshua Conte

October 29, 2017

1 M6L1 Homework Assignment

R studio was configured with the following parameters before beginning the project:

```
# clears the console in RStudio
cat("\014")
```

```
# clears environment
rm(list = ls())

# Load required packages
require(ggplot2)
require(class)
```

1.1 Load Data.

I opened the Wholesale customers Data Set using read.csv2 and downloaded it directly from the UC Irvine Machine Learning Repository.

To format the data, the data is separated by ‘,’, stringsAsFactors = FALSE so that the strings in a data frame will be treated as plain strings and not as factor variables. I set na strings for missing data. Once the data was loaded I added the column names and changed the data types to numeric and finally removed the text data type.

Below is my R code:

```
# Some csv files are really big and take a while to open.  This command checks to
# see if it is already opened, if it is, it does not open it again.
# I also omitted the first column
if (!exists("dfWCD")) {
dfWCD <-
  read.csv2("Wholesale customers data.csv",
    sep = ",",
    stringsAsFactors = FALSE,
    na.strings=c("","NA")
  )
}

# Download directly from site (unreliable from Ecuador)
# if (!exists("dfWCD")) {
# dfWCD <-
#   read.csv2(
#     url(
#       "https://archive.ics.uci.edu/ml/machine-learning-databases/00292/Wholesale customers data.csv"
#     ),
#     sep = ",",
#     stringsAsFactors = FALSE,
#     na.strings=c("","NA")
#   )
# # Add a column so I know which study the data is referring to
# study <- sprintf("study_%s",seq(1:440))
# dfWCD$study<-study
# }

# change 2 to 24 to numeric
```

```
dfWCD[1:8] <- sapply(dfWCD[1:8], as.numeric)

# Print first lines
str(dfWCD)
```

```
## 'data.frame':    440 obs. of  8 variables:
##  $ Channel         : num  2 2 2 1 2 2 2 2 1 2 ...
##  $ Region          : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ Fresh           : num  12669 7057 6353 13265 22615 ...
##  $ Milk            : num  9656 9810 8808 1196 5410 ...
##  $ Grocery         : num  7561 9568 7684 4221 7198 ...
##  $ Frozen          : num  214 1762 2405 6404 3915 ...
##  $ Detergents_Paper: num  2674 3293 3516 507 1777 ...
##  $ Delicassen      : num  1338 1776 7844 1788 5185 ...
```

1.1.1 Understanding the data

The data set refers to clients of a wholesale distributor in Portugal. It includes the annual spending in monetary units (m.u.) on diverse product categories. The data has the following attribute information:

1. FRESH: annual spending (m.u.) on fresh products (Continuous);
2. MILK: annual spending (m.u.) on milk products (Continuous);
3. GROCERY: annual spending (m.u.)on grocery products (Continuous);
4. FROZEN: annual spending (m.u.)on frozen products (Continuous)
5. DETERGENTS_PAPER: annual spending (m.u.) on detergents and paper products (Continuous)
6. DELICATESSEN: annual spending (m.u.)on and delicatessen products (Continuous);
7. CHANNEL: customer channel - 1 = Horeca (Hotel/Restaurant/Cafe) or 2 = Retail
8. REGION: Customers Region - 1= Lisnon 2 = Oporto or 3 = Other (Nominal)

1.2 k-Nearest Neighbors (kNN) in R

A simple supervised learning algorithm is k-Nearest Neighbors algorithm (k-NN). KNN is a non-parametric method used for classification and regression.

In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

I am using the Channel data for classification:

```
head(dfWCD)
```

```
##   Channel Region Fresh Milk Grocery Frozen Detergents_Paper Delicassen
## 1       2      3 12669 9656    7561    214             2674       1338
## 2       2      3  7057 9810    9568   1762             3293       1776
## 3       2      3  6353 8808    7684   2405             3516       7844
## 4       1      3 13265 1196    4221   6404              507       1788
## 5       2      3 22615 5410    7198   3915             1777       5185
## 6       2      3  9413 8259    5126    666             1795       1451
```

```
summary(dfWCD)
```

```
##     Channel          Region          Fresh              Milk      
##  Min.   :1.000   Min.   :1.000   Min.   :     3   Min.   :   55  
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:  3128   1st Qu.: 1533  
##  Median :1.000   Median :3.000   Median :  8504   Median : 3627  
##  Mean   :1.323   Mean   :2.543   Mean   : 12000   Mean   : 5796  
##  3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.: 16934   3rd Qu.: 7190  
##  Max.   :2.000   Max.   :3.000   Max.   :112151   Max.   :73498  
##     Grocery          Frozen        Detergents_Paper    Delicassen     
##  Min.   :    3   Min.   :   25.0   Min.   :    3.0   Min.   :    3.0  
##  1st Qu.: 2153   1st Qu.:  742.2   1st Qu.:  256.8   1st Qu.:  408.2  
##  Median : 4756   Median : 1526.0   Median :  816.5   Median :  965.5  
##  Mean   : 7951   Mean   : 3071.9   Mean   : 2881.5   Mean   : 1524.9  
##  3rd Qu.:10656   3rd Qu.: 3554.2   3rd Qu.: 3922.0   3rd Qu.: 1820.2  
##  Max.   :92780   Max.   :60869.0   Max.   :40827.0   Max.   :47943.0
```

```
length(dfWCD)
```

```
## [1] 8
```

```
names(dfWCD)
```

```
## [1] "Channel"          "Region"           "Fresh"           
## [4] "Milk"             "Grocery"          "Frozen"          
## [7] "Detergents_Paper" "Delicassen"
```

```
table(dfWCD$Channel)
```

```
## 
##   1   2 
## 298 142
```

```
dfWCD$Channel
```

```
##   [1] 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 1 2 1 2 1 2 1 1 2 2 2 1 1 2 1 1 1 1 1 1
##  [36] 2 1 2 2 1 1 1 2 2 2 2 2 2 2 2 1 1 2 2 1 1 2 2 1 1 2 2 2 2 1 2 1 2 1 1
##  [71] 1 1 1 2 2 1 1 2 1 1 1 2 2 1 2 2 2 1 1 1 1 1 2 1 2 1 2 1 1 1 2 2 2 1 1
## [106] 1 2 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
## [141] 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 2 2 2 1 1 2 2 2 2 1 1 1 2 2 1 2 1
## [176] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1 1 1 2 1 1 2 2 1 1 1 2 1 2 1 2
## [211] 1 2 1 1 2 1 2 1 2 1 1 1 1 2 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [246] 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2
## [281] 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 2 1 2 2 2 2 2 2 2 1 1 2 1 1 2 1 1
## [316] 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 1 1 1 1 2 2 1 2 1 1 2 2 1 2
## [351] 1 2 1 2 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 2 1 1 2 1 1 2 1 1 1 1 1
## [386] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 2 2 1 2 1
## [421] 1 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
```

```
length(dfWCD$Channel)
```

```
## [1] 440
```

In this section I shuffle the data to make it a little more random to make sure the training set and predictive set have a good random sample, I don't want to train on all 1's to predict for the 2's:

```
shuff<-runif(nrow(dfWCD))
head(shuff)
```

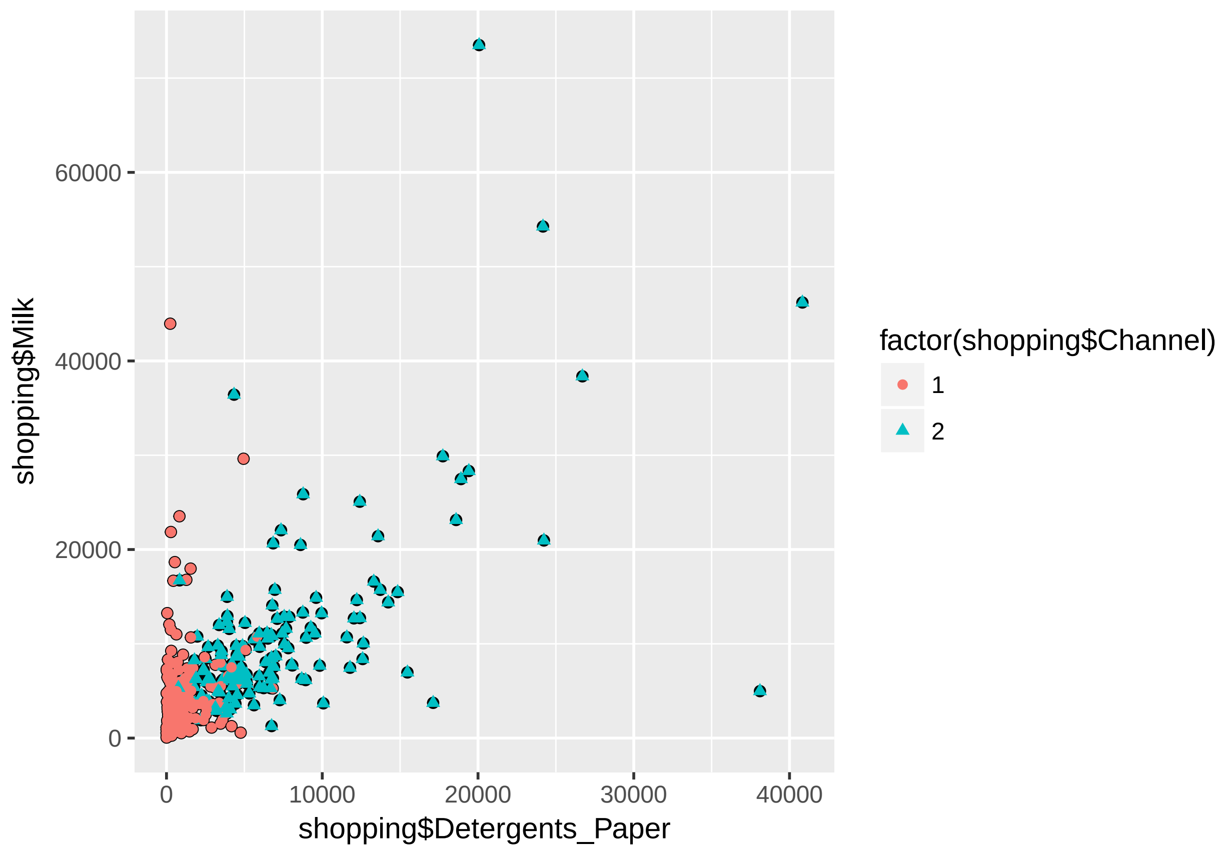
```
## [1] 0.05098633 0.28433827 0.36240784 0.79634407 0.46410232 0.86816866
```

```
shopping<-dfWCD[order(shuff),]
shopping$Channel
```

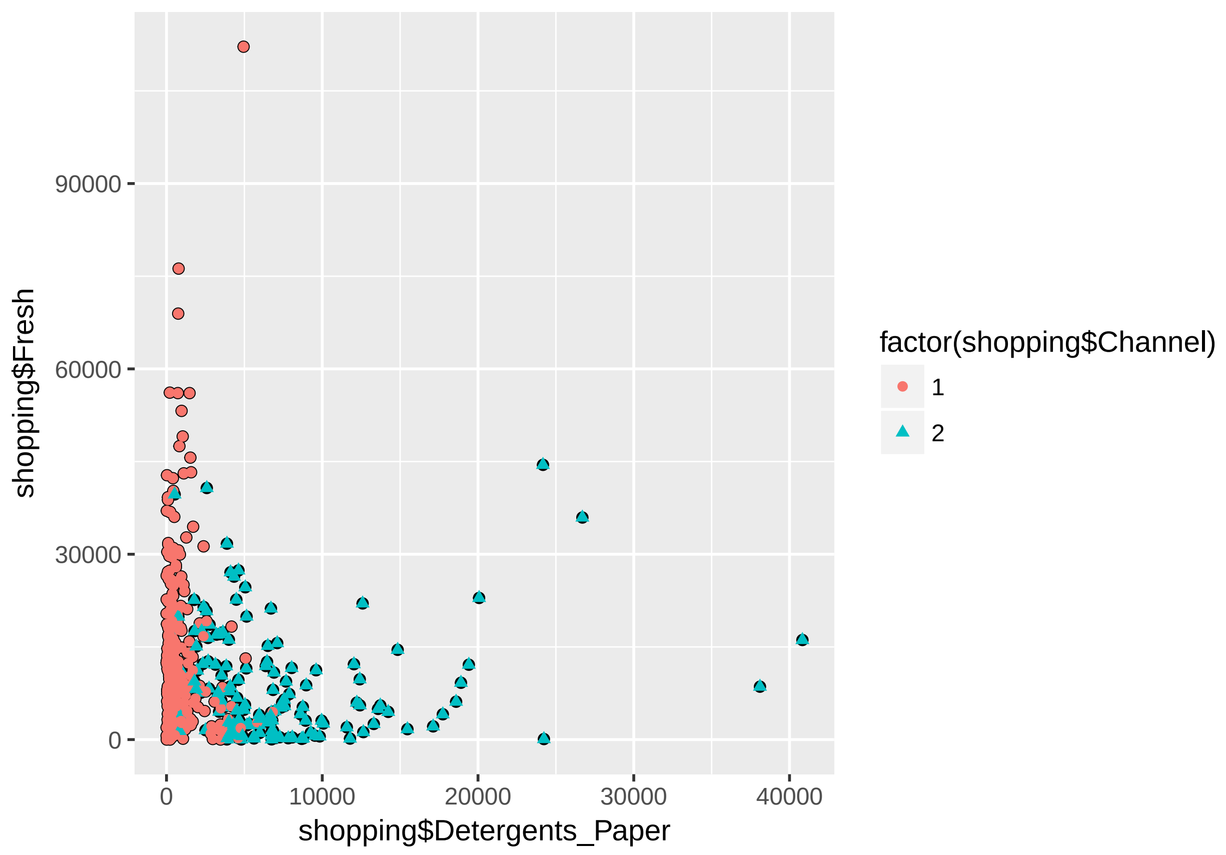
```
##   [1] 1 2 2 2 1 1 1 1 1 1 1 2 1 1 2 2 1 1 2 2 1 1 2 1 2 1 1 2 1 1 1 2 1 1 1
##  [36] 1 2 1 2 1 1 2 2 1 2 1 1 2 2 1 2 1 1 1 1 1 1 1 1 1 2 2 1 2 1 1 1 1 1 2
##  [71] 1 1 1 2 1 2 2 2 2 1 1 1 2 1 1 2 1 2 1 2 1 1 1 2 1 1 1 2 2 2 1 2 1 1 1
## [106] 1 2 2 1 1 1 2 1 1 1 1 2 2 1 1 1 2 1 1 1 1 1 1 1 2 1 2 2 2 1 2 1 2 1 1
## [141] 1 2 1 2 1 1 2 1 1 1 1 2 1 1 1 2 1 1 2 1 1 1 1 1 1 2 1 1 1 2 1 1 1 2 1
## [176] 2 2 1 2 1 2 1 2 1 1 2 1 1 2 1 1 1 1 2 1 1 1 1 2 1 2 1 2 2 1 1 1 1 2 2
## [211] 2 1 1 2 1 2 1 1 1 2 2 1 2 1 2 1 2 1 1 2 1 2 1 2 1 2 1 1 1 1 2 1 1 1 1
## [246] 1 2 1 1 2 1 1 1 2 2 1 2 1 2 1 1 1 1 2 1 1 1 2 1 1 1 1 2 1 1 2 1 1 1 1
## [281] 2 1 1 1 2 1 1 2 1 1 1 1 2 1 2 1 2 1 2 2 1 1 2 1 1 1 2 1 1 1 1 1 2 2 1
## [316] 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 2 1 2 1 1 1
## [351] 1 1 1 1 1 2 1 1 2 1 1 1 2 1 2 1 1 2 1 1 2 1 2 1 1 2 1 2 1 1 1 1 2 1 1
## [386] 2 2 2 1 1 1 1 1 2 1 1 2 1 1 1 2 1 2 1 2 1 2 2 1 1 1 2 1 1 1 2 1 1 1 1
## [421] 1 1 2 1 2 2 1 1 2 1 1 2 1 1 1 1 1 1 1 2
```

This looks at the plotting to see if the knn will do a good job. Here I plot Detergents_Paper vs Milk and Detergents_Paper vs Fresh:

```
qplot(shopping$Detergents_Paper,shopping$Milk,data=shopping)+geom_point(aes(colour = factor(shopping$Cha
```



```
qplot(shopping$Detergents_Paper,shopping$Fresh,data=shopping)+geom_point(aes(colour = factor(shopping$Ch
```



```r
summary(shopping)
```

```
##     Channel          Region          Fresh             Milk      
##  Min.   :1.000   Min.   :1.000   Min.   :     3   Min.   :   55  
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:  3128   1st Qu.: 1533  
##  Median :1.000   Median :3.000   Median :  8504   Median : 3627  
##  Mean   :1.323   Mean   :2.543   Mean   : 12000   Mean   : 5796  
##  3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.: 16934   3rd Qu.: 7190  
##  Max.   :2.000   Max.   :3.000   Max.   :112151   Max.   :73498  
##     Grocery          Frozen         Detergents_Paper    Delicassen     
##  Min.   :    3   Min.   :   25.0   Min.   :    3.0   Min.   :    3.0  
##  1st Qu.: 2153   1st Qu.:  742.2   1st Qu.:  256.8   1st Qu.:  408.2  
##  Median : 4756   Median : 1526.0   Median :  816.5   Median :  965.5  
##  Mean   : 7951   Mean   : 3071.9   Mean   : 2881.5   Mean   : 1524.9  
##  3rd Qu.:10656   3rd Qu.: 3554.2   3rd Qu.: 3922.0   3rd Qu.: 1820.2  
##  Max.   :92780   Max.   :60869.0   Max.   :40827.0   Max.   :47943.0  
```

This algorithm should do a good job since the data is not mixed, the plots look separated.

Here I grab the data that can be analyzed, I no longer need Channel or Regions, and I scale the data to make it all on a comparable scale.

```r
shopping.scaled<-as.data.frame(lapply(shopping[,c(3:8)], scale))
head(shopping.scaled)
```

```
##          Fresh        Milk    Grocery      Frozen Detergents_Paper
## 1 -0.054264243 -0.3666840 -0.6197176  6.57862352       -0.5894671
## 2 -0.747928502 -0.1435246 -0.1959639 -0.02594033        0.5106085
## 3 -0.591848113  0.8393520  0.1466588 -0.08814843        0.1041363
## 4 -0.007297804  0.3057749  1.2517646 -0.41257808        0.7325112
## 5 -0.458460264  0.2681074  0.3015546 -0.38909556        0.0548479
## 6  1.022247655 -0.5413363 -0.5762584 -0.17631090       -0.5179464
##   Delicassen
## 1  0.4159878
## 2 -0.4981623
## 3  0.2440084
## 4 -0.2932055
## 5  0.1578414
## 6 -0.1694512
```

```r
summary(shopping.scaled)
```

```
##      Fresh              Milk            Grocery            Frozen
##  Min.   :-0.9486   Min.   :-0.7779   Min.   :-0.8364   Min.   :-0.62763
##  1st Qu.:-0.7015   1st Qu.:-0.5776   1st Qu.:-0.6101   1st Qu.:-0.47988
##  Median :-0.2764   Median :-0.2939   Median :-0.3363   Median :-0.31844
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.3901   3rd Qu.: 0.1889   3rd Qu.: 0.2846   3rd Qu.: 0.09935
##  Max.   : 7.9187   Max.   : 9.1732   Max.   : 8.9264   Max.   :11.90545
##  Detergents_Paper    Delicassen
##  Min.   :-0.6037   Min.   :-0.5396
##  1st Qu.:-0.5505   1st Qu.:-0.3960
##  Median :-0.4331   Median :-0.1984
##  Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.2182   3rd Qu.: 0.1047
##  Max.   : 7.9586   Max.   :16.4597
```

This also scales the data like above and puts all of it on a scale from 0 to 1.

```r
normalize<- function(x) {
  return((x-min(x))/(max(x)-min(x)))
}
shopping.normalized<-as.data.frame(lapply(shopping[,c(3:8)],normalize))
head(shopping.normalized)
```

```
##        Fresh       Milk    Grocery     Frozen Detergents_Paper  Delicassen
## 1 0.10085780 0.04132456 0.02219300 0.57497863      0.001665687 0.056216103
## 2 0.02263081 0.06375012 0.06559815 0.04800802      0.130144033 0.002440551
## 3 0.04023255 0.16252059 0.10069306 0.04304451      0.082671958 0.046099291
## 4 0.10615437 0.10890078 0.21388922 0.01715864      0.156060161 0.014497288
## 5 0.05527517 0.10511553 0.11655906 0.01903228      0.076915540 0.041030455
## 6 0.22225987 0.02377354 0.02664453 0.03601012      0.010018617 0.021777222
```

```r
summary(shopping.normalized)
```

```
##      Fresh              Milk            Grocery            Frozen
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.02786   1st Qu.:0.02012   1st Qu.:0.02317   1st Qu.:0.01179
##  Median :0.07580   Median :0.04864   Median :0.05122   Median :0.02467
##  Mean   :0.10698   Mean   :0.07817   Mean   :0.08567   Mean   :0.05008
##  3rd Qu.:0.15097   3rd Qu.:0.09715   3rd Qu.:0.11482   3rd Qu.:0.05800
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##  Detergents_Paper     Delicassen
##  Min.   :0.000000   Min.   :0.000000
##  1st Qu.:0.006216   1st Qu.:0.008453
##  Median :0.019927   Median :0.020077
##  Mean   :0.070510   Mean   :0.031745
##  3rd Qu.:0.095997   3rd Qu.:0.037907
##  Max.   :1.000000   Max.   :1.000000
```

```
nrow(shopping)
```

```
## [1] 440
```

This takes the normalized data and trains the first 390 to predict the remaining 50, with a k=4:

```
shopping.normalized.train<-shopping.normalized[1:390,]
shopping.normalized.test<-shopping.normalized[391:440,]
shopping.normalized.train.target<-shopping[1:390,c(1)]
shopping.normalized.test.target<-shopping[391:440,c(1)]
shopping.normalized.test.target
```

```
##  [1] 1 1 1 2 1 1 2 1 1 1 2 1 2 1 2 1 2 2 1 1 1 2 1 1 1 2 1 1 1 1 1 1 2 1 2
## [36] 2 1 1 2 1 1 2 1 1 1 1 1 1 1 2
```

```
k<-4
knn.m1.4<-knn(train = shopping.normalized.train, test = shopping.normalized.test,shopping.normalized.tra
knn.m1.4
```

```
##  [1] 1 1 1 2 1 2 2 1 1 1 2 1 2 1 2 1 2 2 1 1 1 2 1 1 1 2 1 1 1 2 1 2 2 2 2
## [36] 2 1 1 2 1 1 2 1 1 1 1 1 1 1 2
## Levels: 1 2
```

```
length(knn.m1.4)
```

```
## [1] 50
```

```
cm<-table(shopping.normalized.test.target,knn.m1.4)
cm
```

```
##                                knn.m1.4
## shopping.normalized.test.target  1  2
##                               1 31  4
##                               2  0 15
```

The results look like 3 out of 33 were mislabeled as Channel 2 instead of Channel 1 (9.09% error) and 0 out of 17 were mislabeled as Channel 1 instead of Channel 2 (0% error). Overall, I think it did a good job.

1.2.1 Using different k's

This uses k=2:

```
k<-2
knn.m1.2<-knn(train = shopping.normalized.train, test = shopping.normalized.test,shopping.normalized.tra
cm<-table(shopping.normalized.test.target,knn.m1.2)
cm
```

```
##                                knn.m1.2
## shopping.normalized.test.target  1  2
##                               1 32  3
##                               2  2 13
```

Using k = 2 gives a great result of 1 out of 33 were mislabeled as Channel 2 instead of Channel 1 (3.03% error) and 0 out of 17 were mislabeled as Channel 1 instead of Channel 2 (0% error).

This uses k=3:

```
k<-3
knn.m1.3<-knn(train = shopping.normalized.train, test = shopping.normalized.test,shopping.normalized.tra
cm<-table(shopping.normalized.test.target,knn.m1.3)
cm
```

```
##                                knn.m1.3
## shopping.normalized.test.target  1  2
##                               1 32  3
##                               2  0 15
```

Using k = 3 gives a great result of 2 out of 33 were mislabeled as Channel 2 instead of Channel 1 (6.06% error) and 0 out of 17 were mislabeled as Channel 1 instead of Channel 2 (0% error).

This uses k=5:

```
k<-5
knn.m1.5<-knn(train = shopping.normalized.train, test = shopping.normalized.test,shopping.normalized.tra
cm<-table(shopping.normalized.test.target,knn.m1.5)
cm
```

```
##                                knn.m1.5
## shopping.normalized.test.target  1  2
##                               1 30  5
##                               2  1 14
```

Using k = 5 gives the same result as k = 4.

This uses k=6:

```
k<-6
knn.m1.6<-knn(train = shopping.normalized.train, test = shopping.normalized.test,shopping.normalized.tra
cm<-table(shopping.normalized.test.target,knn.m1.6)
cm
```

```
##                                knn.m1.6
## shopping.normalized.test.target  1  2
##                               1 30  5
##                               2  0 15
```

Using k = 6 gives the same result as k = 4.

This uses k=7:

```
k<-7
knn.m1.7<-knn(train = shopping.normalized.train, test = shopping.normalized.test,shopping.normalized.tra
cm<-table(shopping.normalized.test.target,knn.m1.7)
cm
```

```
##                                knn.m1.7
## shopping.normalized.test.target  1  2
##                               1 30  5
##                               2  1 14
```

Using k = 7 gives the same result as k = 3.

This uses k=8:

```
k<-8
knn.m1.8<-knn(train = shopping.normalized.train, test = shopping.normalized.test,shopping.normalized.tra
cm<-table(shopping.normalized.test.target,knn.m1.8)
cm
```

```
##                                  knn.m1.8
## shopping.normalized.test.target  1  2
##                                1 31  4
##                                2  0 15
```

Using k = 8 gives the same result as k = 3.

1.2.2 Using the original data

Does scaling, normalization or leaving the data unscaled make a difference for kNN?

```
shopping.train<-shopping[1:390,]
shopping.test<-shopping[391:440,]
shopping.train.target<-shopping[1:390,c(1)]
shopping.test.target<-shopping[391:440,c(1)]
shopping.test.target
```

```
##  [1] 1 1 1 2 1 1 2 1 1 1 2 1 2 1 2 1 2 2 1 1 1 2 1 1 1 2 1 1 1 1 1 1 2 1 2
## [36] 2 1 1 2 1 1 2 1 1 1 1 1 1 1 2
```

```
k<-3
knn.m1.4<-knn(train = shopping.train, test = shopping.test,shopping.train.target,k)
knn.m1.4
```

```
##  [1] 1 2 1 2 1 2 2 1 1 1 2 1 2 1 2 1 2 2 1 1 1 2 1 1 1 2 1 1 2 2 1 2 2 2 2
## [36] 2 1 1 2 1 1 2 1 1 1 1 1 1 1 2
## Levels: 1 2
```

```
length(knn.m1.4)
```

```
## [1] 50
```

```
cm<-table(shopping.test.target,knn.m1.4)
cm
```

```
##                       knn.m1.4
## shopping.test.target  1  2
##                     1 29  6
##                     2  0 15
```

No, because the data was already to scale.

1.3 Questions

1. Does the k for kNN make a difference? Try for a range of values of k.
   - Yes, the k value makes a difference. I ran KNN with a k value from 2 through 8, 2 was the best followed by 3. A k value of 4, 5, and 6 gave the worst and a k value of 7 and 8 were the same of a k value of 3.
2. Does scaling, normalization or leaving the data unscaled make a difference for kNN? Why or Why not?
   - For this dataset it does not matter because all of the data used were at a similar scale. If I used a different dataset that had mixed data at different scales, like the wine data used in the class lecture, it would make a difference. This is because distance metrics that are not scale invariant, attributes with larger absolute ranges will dominate.