Support Vector Machine

Chandan Gowda Ashwath

9/9/2020

1. Introduction:

This assignment will look at a data "juice.csv" which contains purchase information for Citrus Hill or Minute Maid orange juice.

This assignment is interested in predicting whether the customer purchased Citrus Hill or Minute Maid Orange Juice. Support Vector Machines will be used for this prediction.

We will be using three kernels i.e., linear, radial, and polynomial.

2. Data and Analysis:

Let us load the data and perform some exploratory analysis.

```
if(!require("pacman"))
install.packages("pacman")
```

```
## Loading required package: pacman
```

```
## Warning: package 'pacman' was built under R version 3.6.3
```

```
pacman::p_load(e1071, ggplot2, caret, rmarkdown, corrplot, knitr)
search()
```

```
##  [1] ".GlobalEnv"        "package:knitr"     "package:corrplot"
##  [4] "package:rmarkdown" "package:caret"     "package:lattice"
##  [7] "package:ggplot2"   "package:e1071"     "package:pacman"
## [10] "package:stats"     "package:graphics"  "package:grDevices"
## [13] "package:utils"     "package:datasets"  "package:methods"
## [16] "Autoloads"         "package:base"
```

```
theme_set(theme_classic())
options(digits = 3)
data<-read.csv("juice.csv")
```

Let us take a look at the summary and Structure of the "JUICES" data.

From the summary function, we can see that some of the variables have 5 number summaries in the form of "Min, 1st Qu, Median, Mean, 3rd Qu, Max", when these variables should be more categorical in nature. These are not so meaningful for categorical variables.

StoreID, SpecialCH, SpecialMM and Store are examples of the categorical variables with summaries.

```
summary(data)
```

```
##  Purchase WeekofPurchase    StoreID        PriceCH        PriceMM
##  CH:610   Min.   :227    Min.   :1.00   Min.   :1.69   Min.   :1.69
##  MM:390   1st Qu.:239    1st Qu.:2.00   1st Qu.:1.79   1st Qu.:1.99
##           Median :256    Median :3.00   Median :1.86   Median :2.09
##           Mean   :254    Mean   :3.98   Mean   :1.87   Mean   :2.08
##           3rd Qu.:268    3rd Qu.:7.00   3rd Qu.:1.99   3rd Qu.:2.18
##           Max.   :278    Max.   :7.00   Max.   :2.09   Max.   :2.29
##      DiscCH          DiscMM        SpecialCH       SpecialMM        LoyalCH
##  Min.   :0.00   Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:0.00   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.320
##  Median :0.00   Median :0.000   Median :0.000   Median :0.000   Median :0.591
##  Mean   :0.05   Mean   :0.124   Mean   :0.142   Mean   :0.163   Mean   :0.562
##  3rd Qu.:0.00   3rd Qu.:0.240   3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.844
##  Max.   :0.50   Max.   :0.800   Max.   :1.000   Max.   :1.000   Max.   :1.000
##   SalePriceMM    SalePriceCH     PriceDiff      Store7      PctDiscMM
##  Min.   :1.19   Min.   :1.39   Min.   :-0.670   No :665   Min.   :0.000
##  1st Qu.:1.69   1st Qu.:1.75   1st Qu.: 0.000   Yes:335   1st Qu.:0.000
##  Median :2.09   Median :1.86   Median : 0.230             Median :0.000
##  Mean   :1.96   Mean   :1.82   Mean   : 0.143             Mean   :0.060
##  3rd Qu.:2.13   3rd Qu.:1.89   3rd Qu.: 0.300             3rd Qu.:0.113
##  Max.   :2.29   Max.   :2.09   Max.   : 0.640             Max.   :0.402
##    PctDiscCH      ListPriceDiff       STORE
##  Min.   :0.0000   Min.   :0.000   Min.   :0.00
##  1st Qu.:0.0000   1st Qu.:0.140   1st Qu.:0.00
##  Median :0.0000   Median :0.240   Median :2.00
##  Mean   :0.0263   Mean   :0.217   Mean   :1.63
##  3rd Qu.:0.0000   3rd Qu.:0.300   3rd Qu.:3.00
##  Max.   :0.2527   Max.   :0.440   Max.   :4.00
```

```
str(data)
```

```
## 'data.frame':    1000 obs. of  18 variables:
##  $ Purchase      : Factor w/ 2 levels "CH","MM": 2 1 2 1 1 2 1 1 1 1 ...
##  $ WeekofPurchase: int  237 258 242 271 276 240 248 270 266 274 ...
##  $ StoreID       : int  2 7 3 2 2 1 3 1 2 7 ...
##  $ PriceCH       : num  1.75 1.86 1.99 1.86 1.99 1.75 1.99 1.86 1.86 1.86 ...
##  $ PriceMM       : num  1.99 2.18 2.23 2.18 2.18 1.99 2.23 2.18 2.18 2.13 ...
##  $ DiscCH        : num  0 0 0 0 0 0 0 0 0 0.47 ...
##  $ DiscMM        : num  0 0 0 0.06 0 0.3 0 0 0 0.54 ...
##  $ SpecialCH     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ SpecialMM     : int  0 0 0 0 1 1 0 0 0 0 ...
##  $ LoyalCH       : num  0.4 0.90814 0.00721 0.78839 0.97251 ...
##  $ SalePriceMM   : num  1.99 2.18 2.23 2.12 2.18 1.69 2.23 2.18 2.18 1.59 ...
##  $ SalePriceCH   : num  1.75 1.86 1.99 1.86 1.99 1.75 1.99 1.86 1.86 1.39 ...
##  $ PriceDiff     : num  0.24 0.32 0.24 0.26 0.19 -0.06 0.24 0.32 0.32 0.2 ...
##  $ Store7        : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 1 1 1 2 ...
##  $ PctDiscMM     : num  0 0 0 0.0275 0 ...
##  $ PctDiscCH     : num  0 0 0 0 0 ...
##  $ ListPriceDiff : num  0.24 0.32 0.24 0.32 0.19 0.24 0.24 0.32 0.32 0.27 ...
##  $ STORE         : int  2 0 3 2 2 1 3 1 2 0 ...
```

So, let us first convert these variables to categories.

From the summary data, There are three store variables i.e, StoreID, Store7 and STORE and all these variables are related. StoreID and STORE contains same number of observations except for store 7 values converted as 0.

Also, the SpecialCH and SpecialMM show that most purchases did not include a special value on either juice brand.

```
data1<-data
cvar = c("StoreID","SpecialCH","SpecialMM","STORE")
data1[cvar] = lapply(data1[cvar], as.factor)
summary(data1)
```

```
##  Purchase WeekofPurchase StoreID    PriceCH        PriceMM         DiscCH
##  CH:610   Min.   :227    1:146   Min.   :1.69   Min.   :1.69   Min.   :0.00
##  MM:390   1st Qu.:239    2:204   1st Qu.:1.79   1st Qu.:1.99   1st Qu.:0.00
##           Median :256    3:182   Median :1.86   Median :2.09   Median :0.00
##           Mean   :254    4:133   Mean   :1.87   Mean   :2.08   Mean   :0.05
##           3rd Qu.:268    7:335   3rd Qu.:1.99   3rd Qu.:2.18   3rd Qu.:0.00
##           Max.   :278            Max.   :2.09   Max.   :2.29   Max.   :0.50
##      DiscMM      SpecialCH SpecialMM    LoyalCH       SalePriceMM
##  Min.   :0.000   0:858     0:837     Min.   :0.000   Min.   :1.19
##  1st Qu.:0.000   1:142     1:163     1st Qu.:0.320   1st Qu.:1.69
##  Median :0.000                       Median :0.591   Median :2.09
##  Mean   :0.124                       Mean   :0.562   Mean   :1.96
##  3rd Qu.:0.240                       3rd Qu.:0.844   3rd Qu.:2.13
##  Max.   :0.800                       Max.   :1.000   Max.   :2.29
##   SalePriceCH     PriceDiff      Store7      PctDiscMM       PctDiscCH
##  Min.   :1.39   Min.   :-0.670   No :665   Min.   :0.000   Min.   :0.0000
##  1st Qu.:1.75   1st Qu.: 0.000   Yes:335   1st Qu.:0.000   1st Qu.:0.0000
##  Median :1.86   Median : 0.230             Median :0.000   Median :0.0000
##  Mean   :1.82   Mean   : 0.143             Mean   :0.060   Mean   :0.0263
##  3rd Qu.:1.89   3rd Qu.: 0.300             3rd Qu.:0.113   3rd Qu.:0.0000
##  Max.   :2.09   Max.   : 0.640             Max.   :0.402   Max.   :0.2527
##  ListPriceDiff   STORE
##  Min.   :0.000   0:335
##  1st Qu.:0.140   1:146
##  Median :0.240   2:204
##  Mean   :0.217   3:182
##  3rd Qu.:0.300   4:133
##  Max.   :0.440
```

```
str(data1)
```

```
## 'data.frame':    1000 obs. of  18 variables:
##  $ Purchase      : Factor w/ 2 levels "CH","MM": 2 1 2 1 1 2 1 1 1 1 ...
##  $ WeekofPurchase: int  237 258 242 271 276 240 248 270 266 274 ...
##  $ StoreID       : Factor w/ 5 levels "1","2","3","4",..: 2 5 3 2 2 1 3 1 2 5 ...
##  $ PriceCH       : num  1.75 1.86 1.99 1.86 1.99 1.75 1.99 1.86 1.86 1.86 ...
##  $ PriceMM       : num  1.99 2.18 2.23 2.18 2.18 1.99 2.23 2.18 2.18 2.13 ...
##  $ DiscCH        : num  0 0 0 0 0 0 0 0 0 0.47 ...
##  $ DiscMM        : num  0 0 0 0.06 0 0.3 0 0 0 0.54 ...
##  $ SpecialCH     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
```

```
##  $ SpecialMM     : Factor w/ 2 levels "0","1": 1 1 1 1 2 2 1 1 1 1 ...
##  $ LoyalCH       : num  0.4 0.90814 0.00721 0.78839 0.97251 ...
##  $ SalePriceMM   : num  1.99 2.18 2.23 2.12 2.18 1.69 2.23 2.18 2.18 1.59 ...
##  $ SalePriceCH   : num  1.75 1.86 1.99 1.86 1.99 1.75 1.99 1.86 1.86 1.39 ...
##  $ PriceDiff     : num  0.24 0.32 0.24 0.26 0.19 -0.06 0.24 0.32 0.32 0.2 ...
##  $ Store7        : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 1 1 1 2 ...
##  $ PctDiscMM     : num  0 0 0 0.0275 0 ...
##  $ PctDiscCH     : num  0 0 0 0 0 ...
##  $ ListPriceDiff : num  0.24 0.32 0.24 0.32 0.19 0.24 0.24 0.32 0.32 0.27 ...
##  $ STORE         : Factor w/ 5 levels "0","1","2","3",..: 3 1 4 3 3 2 4 2 3 1 ...
```

Let us see the correlation among the numerical variables

One can see almost perfect correlation of arround 0.99 between two pairs of variables i.e., PctDiscMM and DiscMM, and PctDiscCH and DiscCH. The two pairs of variables show almost a straight line. These two pairs also have an overall correlation value of around 0.99.

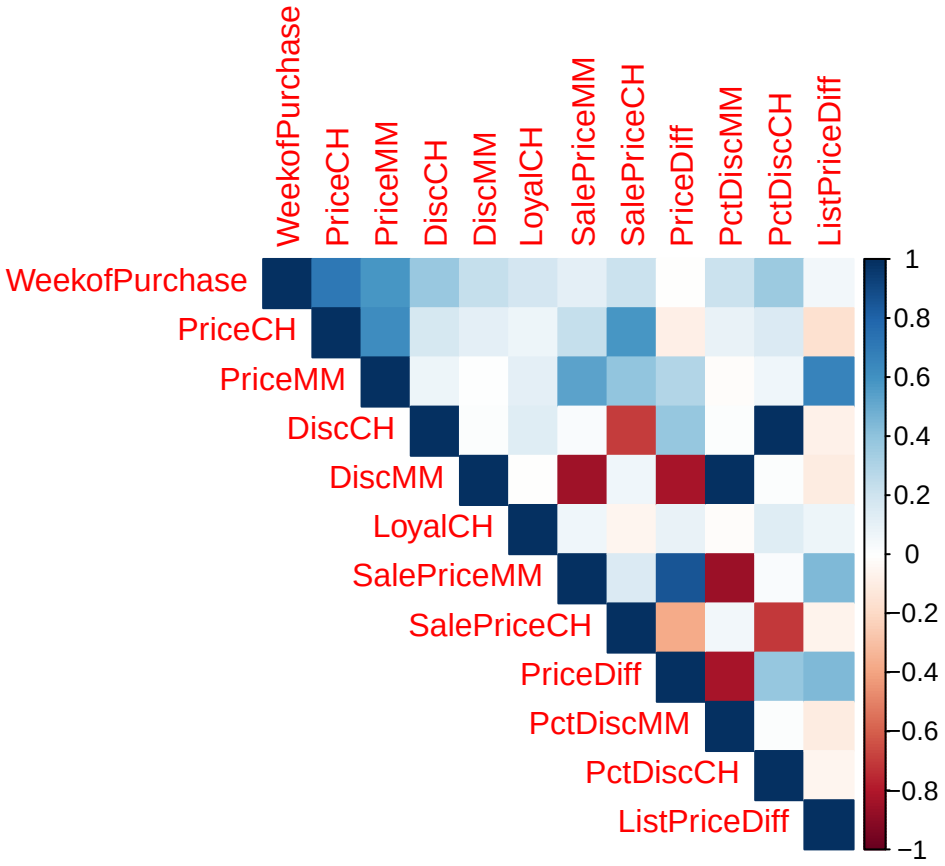
There are also a couple of other highly correlated pairs with > 0.8 in magnitude as well, such as PctDicMM and PriceDiff. These show a general linear trend between the two variables that is easy to distinguish.

```
data2 <- data1[, -c(1, 3, 8, 9, 14, 18)]
cor(data2)
```

```
##                 WeekofPurchase PriceCH  PriceMM  DiscCH   DiscMM  LoyalCH
## WeekofPurchase         1.00000  0.7143  0.58339  0.3746  0.23744  0.18577
## PriceCH                0.71431  1.0000  0.62382  0.1707  0.11347  0.07155
## PriceMM                0.58339  0.6238  1.00000  0.0732  0.00592  0.11031
## DiscCH                 0.37464  0.1707  0.07315  1.0000  0.01902  0.13196
## DiscMM                 0.23744  0.1135  0.00592  0.0190  1.00000 -0.00951
## LoyalCH                0.18577  0.0716  0.11031  0.1320 -0.00951  1.00000
## SalePriceMM            0.11236  0.2390  0.53219  0.0232 -0.84346  0.06731
## SalePriceCH            0.21173  0.5871  0.39361 -0.6974  0.06691 -0.05638
## PriceDiff             -0.00524 -0.0825  0.29334  0.3860 -0.82565  0.09254
## PctDiscMM              0.21843  0.0965 -0.01411  0.0152  0.99880 -0.01171
## PctDiscCH              0.36410  0.1532  0.06833  0.9990  0.01953  0.13131
## ListPriceDiff          0.05629 -0.1646  0.66821 -0.0702 -0.10055  0.07110
##                 SalePriceMM SalePriceCH PriceDiff PctDiscMM PctDiscCH
## WeekofPurchase       0.1124      0.2117  -0.00524    0.2184    0.3641
## PriceCH              0.2390      0.5871  -0.08251    0.0965    0.1532
## PriceMM              0.5322      0.3936   0.29334   -0.0141    0.0683
## DiscCH               0.0232     -0.6974   0.38599    0.0152    0.9990
## DiscMM              -0.8435      0.0669  -0.82565    0.9988    0.0195
## LoyalCH              0.0673     -0.0564   0.09254   -0.0117    0.1313
## SalePriceMM          1.0000      0.1548   0.85661   -0.8532    0.0202
## SalePriceCH          0.1548      1.0000  -0.37714    0.0576   -0.7094
## PriceDiff            0.8566     -0.3771   1.00000   -0.8299    0.3894
## PctDiscMM           -0.8532      0.0576  -0.82995    1.0000    0.0158
## PctDiscCH            0.0202     -0.7094   0.38939    0.0158    1.0000
## ListPriceDiff        0.4441     -0.0621   0.44873   -0.1096   -0.0596
##                 ListPriceDiff
## WeekofPurchase         0.0563
## PriceCH               -0.1646
## PriceMM                0.6682
## DiscCH                -0.0702
```

```
## DiscMM               -0.1005
## LoyalCH               0.0711
## SalePriceMM           0.4441
## SalePriceCH          -0.0621
## PriceDiff             0.4487
## PctDiscMM            -0.1096
## PctDiscCH            -0.0596
## ListPriceDiff         1.0000
```

```
corrplot(cor(data2), method = "color", type = "upper", tl.srt = 90)
```



3. Data Partition

```
set.seed(123)
trainindex <- createDataPartition(data1$Purchase, p=0.8, list= FALSE)
juice_train <- data1[trainindex, ]
juice_test <- data1[-trainindex, ]
dim(juice_train)
```

```
## [1] 800  18
```

```
dim(juice_test)
```

```
## [1] 200  18
```

4. SVM Model

a. Kernel = Linear

1. Cost = 0.01 (default)

```
svm1 <- svm(Purchase~., data=juice_train, kernel = "linear", cost=0.01)
summary(svm1)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = juice_train, kernel = "linear",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  448
##
##  ( 223 225 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Training error rate:

```
train.pred = predict(svm1, juice_train)
x<-table(juice_train$Purchase, train.pred)
x
```

```
##     train.pred
##       CH  MM
##   CH 436  52
##   MM  82 230
```

```
a.linear<-(x[2] + x[3])/(x[1] + x[2] + x[3] + x[4])
a.linear
```

```
## [1] 0.168
```

Test error rate:

```
test.pred = predict(svm1, juice_test)
x<-table(juice_test$Purchase, test.pred)
x
```

```
##     test.pred
##       CH  MM
##   CH 111  11
##   MM  22  56
```

```
b.linear<-(x[2] + x[3])/(x[1] + x[2] + x[3] + x[4])
b.linear
```

```
## [1] 0.165
```

The training error rate is:

```
a.linear<-a.linear*100
capture.output(cat(a.linear, '%'))
```

```
## [1] "16.8 %"
```

The test error rate is:

```
b.linear<-b.linear*100
capture.output(cat(b.linear, '%'))
```

```
## [1] "16.5 %"
```

Tuning to select the best cost parameter

```
set.seed(123)
tune.out = tune(svm, Purchase ~ ., data = juice_train, kernel = "linear", ranges = list(cost = 10^seq(-
    1, by = 0.25)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost
##  0.562
##
## - best performance: 0.17
##
## - Detailed performance results:
##       cost error dispersion
## 1   0.0100 0.186     0.0419
## 2   0.0178 0.180     0.0417
## 3   0.0316 0.179     0.0354
## 4   0.0562 0.174     0.0291
```

```
## 5   0.1000 0.175     0.0333
## 6   0.1778 0.176     0.0330
## 7   0.3162 0.175     0.0312
## 8   0.5623 0.170     0.0271
## 9   1.0000 0.173     0.0262
## 10  1.7783 0.172     0.0269
## 11  3.1623 0.174     0.0224
## 12  5.6234 0.175     0.0212
## 13 10.0000 0.179     0.0250
```

Tuning shows that optimal cost is :

```
tune.out$best.parameters$cost
```

```
## [1] 0.562
```

2. SVM with best cost for kernel = linear

```
svm1_bestcost <- svm(Purchase~., data=juice_train, kernel = "linear", cost=tune.out$best.parameters$cos
summary(svm1_bestcost)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = juice_train, kernel = "linear",
##     cost = tune.out$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.562
##
## Number of Support Vectors:  344
##
##  ( 171 173 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Training error rate:

```
train.pred = predict(svm1_bestcost, juice_train)
x<-table(juice_train$Purchase, train.pred)
x
```

```
##     train.pred
##       CH  MM
##   CH 429  59
##   MM  74 238
```

```
a.linear.best<-(x[2] + x[3])/(x[1] + x[2] + x[3] + x[4])
a.linear.best
```

```
## [1] 0.166
```

```
 Test error rate:
```

```
test.pred = predict(svm1_bestcost, juice_test)
x<-table(juice_test$Purchase, test.pred)
x
```

```
##     test.pred
##       CH  MM
##   CH 109  13
##   MM  21  57
```

```
b.linear.best<-(x[2] + x[3])/(x[1] + x[2] + x[3] + x[4])
b.linear.best
```

```
## [1] 0.17
```

```
The training error rate is:
```

```
a.linear.best<-a.linear.best*100
capture.output(cat(a.linear.best, '%'))
```

```
## [1] "16.6 %"
```

```
 The test error rate is:
```

```
b.linear.best<-b.linear.best*100
capture.output(cat(b.linear.best, '%'))
```

```
## [1] "17 %"
```

The training error decreases to 16.6% but test error slightly increases from 16.5% to 17% by using best cost.

a. Kernel = radial

1. Cost = 0.01 (default)

```r
set.seed(123)
svm2 <- svm(Purchase~., data=juice_train, kernel = "radial", cost=0.01)
summary(svm2)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = juice_train, kernel = "radial",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##
## Number of Support Vectors:  626
##
##  ( 312 314 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Training error rate:

```r
train.pred = predict(svm2, juice_train)
x<-table(juice_train$Purchase, train.pred)
x
```

```
##     train.pred
##       CH  MM
##   CH 488   0
##   MM 312   0
```

```r
a.radial<-(x[2] + x[3])/(x[1] + x[2] + x[3] + x[4])
a.radial
```

```
## [1] 0.39
```

Test error rate:

```r
test.pred = predict(svm2, juice_test)
x<-table(juice_test$Purchase, test.pred)
x
```

```
##     test.pred
##       CH  MM
##   CH 122   0
##   MM  78   0
```

```
b.radial<-(x[2] + x[3])/(x[1] + x[2] + x[3] + x[4])
b.radial
```

```
## [1] 0.39
```

The training error rate is:

```
a.radial<-a.radial*100
capture.output(cat(a.radial, '%'))
```

```
## [1] "39 %"
```

The test error rate is:

```
b.radial<-b.radial*100
capture.output(cat(b.radial, '%'))
```

```
## [1] "39 %"
```

Tuning to select the best cost parameter

```
set.seed(123)
tune.out = tune(svm, Purchase ~ ., data = juice_train, kernel = "radial", ranges = list(cost = 10^seq(-
    1, by = 0.25)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost
##  0.562
##
## - best performance: 0.176
##
## - Detailed performance results:
##       cost error dispersion
## 1   0.0100 0.390     0.0642
## 2   0.0178 0.390     0.0642
## 3   0.0316 0.362     0.0757
## 4   0.0562 0.207     0.0401
## 5   0.1000 0.186     0.0458
## 6   0.1778 0.188     0.0391
## 7   0.3162 0.180     0.0364
## 8   0.5623 0.176     0.0365
## 9   1.0000 0.184     0.0404
```

```
## 10  1.7783 0.185     0.0394
## 11  3.1623 0.184     0.0363
## 12  5.6234 0.188     0.0358
## 13 10.0000 0.195     0.0355
```

Tuning shows that optimal cost is :

```
tune.out$best.parameters$cost
```

```
## [1] 0.562
```

2. SVM with best cost for kernel = radial

```
svm2_bestcost <- svm(Purchase~., data=juice_train, kernel = "radial", cost=tune.out$best.parameters$cos
summary(svm2_bestcost)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = juice_train, kernel = "radial",
##     cost = tune.out$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.562
##
## Number of Support Vectors:  412
##
##  ( 205 207 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Training error rate:

```
train.pred = predict(svm2_bestcost, juice_train)
x<-table(juice_train$Purchase, train.pred)
x
```

```
##     train.pred
##       CH  MM
##   CH 438  50
##   MM  80 232
```

```
a.radial.best<-(x[2] + x[3])/(x[1] + x[2] + x[3] + x[4])
a.radial.best
```

```
## [1] 0.163
```

Test error rate:

```
test.pred = predict(svm2_bestcost, juice_test)
x<-table(juice_test$Purchase, test.pred)
x
```

```
##     test.pred
##       CH  MM
##   CH 113   9
##   MM  22  56
```

```
b.radial.best<-(x[2] + x[3])/(x[1] + x[2] + x[3] + x[4])
b.radial.best
```

```
## [1] 0.155
```

The training error rate is:

```
a.radial.best<-a.radial.best*100
capture.output(cat(a.radial.best, '%'))
```

```
## [1] "16.2 %"
```

The test error rate is:

```
b.radial.best<-b.radial.best*100
capture.output(cat(b.radial.best, '%'))
```

```
## [1] "15.5 %"
```

The training error decreases to 16.2% and test error decreases to 15.5% by using best cost.

a. Kernel = Poly
1. Cost = 0.01 (default)

```
svm3 <- svm(Purchase~., data=juice_train, kernel = "poly", cost=0.01, degree=2)
summary(svm3)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = juice_train, kernel = "poly",
##     cost = 0.01, degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.01
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  626
##
##  ( 312 314 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Training error rate:

```
train.pred = predict(svm3, juice_train)
x<-table(juice_train$Purchase, train.pred)
x
```

```
##     train.pred
##       CH  MM
##   CH 488   0
##   MM 312   0
```

```
a.poly<-(x[2] + x[3])/(x[1] + x[2] + x[3] + x[4])
a.poly
```

```
## [1] 0.39
```

Test error rate:

```
test.pred = predict(svm3, juice_test)
x<-table(juice_test$Purchase, test.pred)
x
```

```
##     test.pred
##       CH  MM
##   CH 122   0
##   MM  78   0
```

```
b.poly<-(x[2] + x[3])/(x[1] + x[2] + x[3] + x[4])
b.poly
```

```
## [1] 0.39
```

The training error rate is:

```
a.poly<-a.poly*100
capture.output(cat(a.poly, '%'))
```

```
## [1] "39 %"
```

The test error rate is:

```
b.poly<-b.poly*100
capture.output(cat(b.poly, '%'))
```

```
## [1] "39 %"
```

Tuning to select the best cost parameter

```
set.seed(123)
tune.out = tune(svm, Purchase ~ ., data = juice_train, kernel = "poly", degree= 2, ranges = list(cost =
    1, by = 0.25)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  5.62
##
## - best performance: 0.191
##
## - Detailed performance results:
##       cost error dispersion
## 1   0.0100 0.390     0.0642
## 2   0.0178 0.390     0.0642
## 3   0.0316 0.386     0.0639
## 4   0.0562 0.374     0.0630
## 5   0.1000 0.359     0.0483
## 6   0.1778 0.329     0.0574
## 7   0.3162 0.301     0.0649
## 8   0.5623 0.269     0.0563
## 9   1.0000 0.228     0.0489
```

```
## 10  1.7783 0.205     0.0392
## 11  3.1623 0.194     0.0461
## 12  5.6234 0.191     0.0453
## 13 10.0000 0.195     0.0413
```

Tuning shows that optimal cost is :

```
tune.out$best.parameters$cost
```

```
## [1] 5.62
```

2. SVM with best cost for kernel = poly

```
svm3_bestcost <- svm(Purchase~., data=juice_train, kernel = "poly", degree = 2, cost=tune.out$best.para
summary(svm3_bestcost)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = juice_train, kernel = "poly",
##     degree = 2, cost = tune.out$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  5.62
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  404
##
##  ( 197 207 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Training error rate:

```
train.pred = predict(svm3_bestcost, juice_train)
x<-table(juice_train$Purchase, train.pred)
x
```

```
##     train.pred
##       CH  MM
##   CH 453  35
##   MM 103 209
```

```
a.poly.best<-(x[2] + x[3])/(x[1] + x[2] + x[3] + x[4])
a.poly.best
```

```
## [1] 0.172
```

Test error rate:

```
test.pred = predict(svm3_bestcost, juice_test)
x<-table(juice_test$Purchase, test.pred)
x
```

```
##     test.pred
##       CH  MM
##   CH 115   7
##   MM  27  51
```

```
b.poly.best<-(x[2] + x[3])/(x[1] + x[2] + x[3] + x[4])
b.poly.best
```

```
## [1] 0.17
```

The training error rate is:

```
a.poly.best<-a.poly.best*100
capture.output(cat(a.poly.best, '%'))
```

```
## [1] "17.2 %"
```

The test error rate is:

```
b.poly.best<-b.poly.best*100
capture.output(cat(b.poly.best, '%'))
```

```
## [1] "17 %"
```

The training error decreases to 17.2% and test error decreases from 39% to 17% by using best cost.

```
rowLabels = c("Linear Kernel, Cost = 0.01",
              paste("Linear Kernel, with best cost"),
              "Radial Kernel, Cost = 0.01",
              paste("Radial Kernel, with best Cost"),
              "Polynomial Kernel, Degree = 2, Cost = 0.01",
              paste("Polynomial Kernel, Degree = 2, with best Cost"))

trainingErrorRate = c(a.linear,
                  a.linear.best,
                  a.radial,
                  a.radial.best,
```

```
                  a.poly,
                  a.poly.best)

testingErrorRate = c(b.linear,
                  b.linear.best,
                  b.radial,
                  b.radial.best,
                  b.poly,
                  b.poly.best)

df = data.frame(trainingErrorRate, testingErrorRate, row.names = rowLabels)
colnames(df) = c("Training Error Rate", "Testing Error Rate")
kable(df, format = 'markdown')
```

	Training Error Rate	Testing Error Rate
Linear Kernel, Cost = 0.01	16.8	16.5
Linear Kernel, with best cost	16.6	17.0
Radial Kernel, Cost = 0.01	39.0	39.0
Radial Kernel, with best Cost	16.2	15.5
Polynomial Kernel, Degree = 2, Cost = 0.01	39.0	39.0
Polynomial Kernel, Degree = 2, with best Cost	17.2	17.0

From the above table, overall, the Radial Kernel with Best Cost parameter of Cost = 0.562 seems to be producing minimum misclassification error on both training and testing data.

Explanation: From the Table, the SVM with a Radial Kernel and a cost of 0.562 seems to give the best results. It had the best train and test error rate compared to the other SVMs.

Looking at the other information in the table, the Radial and Poly SVMs for default cost = 0.01 had similar results. In other words, the SVM with a radial kernel of deafault cost = 0.01 had similar overall results to the SVM with a polynomial kernel of default cost = 0.01 and it should be noted that the SVMs with radial and polynomial kernels with default cost = 0.01 produced unreliable results without proper classification.