BUAN6357.501.Assignment1_Vo

Import data :

The "juice.csv" data contains purchase information for Citrus Hill or Minute Maid orange juice. A description of the variables follows.

1. Purchase: A factor with levels CH and MM indicating whether the customer purchased Citrus Hill or Minute Maid Orange Juice
2. WeekofPurchase: Week of purchase
3. StoreID: Store ID
4. PriceCH: Price charged for CH
5. PriceMM: Price charged for MM
6. DiscCH: Discount offered for CH
7. DiscMM: Discount offered for MM
8. SpecialCH: Indicator of special on CH
9. SpecialMM: Indicator of special on MM
10. LoyalCH: Customer brand loyalty for CH
11. SalePriceMM: Sale price for MM
12. SalePriceCH: Sale price for CH
13. PriceDiff: Sale price of MM less sale price of CH
14. Store7: A factor with levels No and Yes indicating whether the sale is at Store 7
15. PctDiscMM: Percentage discount for MM
16. PctDiscCH: Percentage discount for CH
17. ListPriceDiff: List price of MM less list price of CH
18. STORE: Which of 5 possible stores the sale occurred at'

```
juice <- read.csv('juice.csv')
head(juice,5)
```

```
##   Purchase WeekofPurchase StoreID PriceCH PriceMM DiscCH DiscMM SpecialCH
## 1       MM            237       2    1.75    1.99      0   0.00         0
## 2       CH            258       7    1.86    2.18      0   0.00         0
## 3       MM            242       3    1.99    2.23      0   0.00         0
## 4       CH            271       2    1.86    2.18      0   0.06         0
## 5       CH            276       2    1.99    2.18      0   0.00         0
##   SpecialMM LoyalCH SalePriceMM SalePriceCH PriceDiff Store7 PctDiscMM
## 1         0 0.40000        1.99        1.75      0.24     No    0.0000
## 2         0 0.90814        2.18        1.86      0.32    Yes    0.0000
## 3         0 0.00721        2.23        1.99      0.24     No    0.0000
## 4         0 0.78839        2.12        1.86      0.26     No    0.0275
## 5         1 0.97251        2.18        1.99      0.19     No    0.0000
##   PctDiscCH ListPriceDiff STORE
## 1         0          0.24     2
## 2         0          0.32     0
## 3         0          0.24     3
## 4         0          0.32     2
## 5         0          0.19     2
```

Explore Data

```
colSums(is.na(juice))
```

```
##       Purchase WeekofPurchase        StoreID        PriceCH        PriceMM
##              0              0              0              0              0
##         DiscCH         DiscMM      SpecialCH      SpecialMM        LoyalCH
##              0              0              0              0              0
##    SalePriceMM    SalePriceCH      PriceDiff         Store7      PctDiscMM
##              0              0              0              0              0
##      PctDiscCH  ListPriceDiff          STORE
##              0              0              0
```

```
summary(juice)
```

```
##  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
```

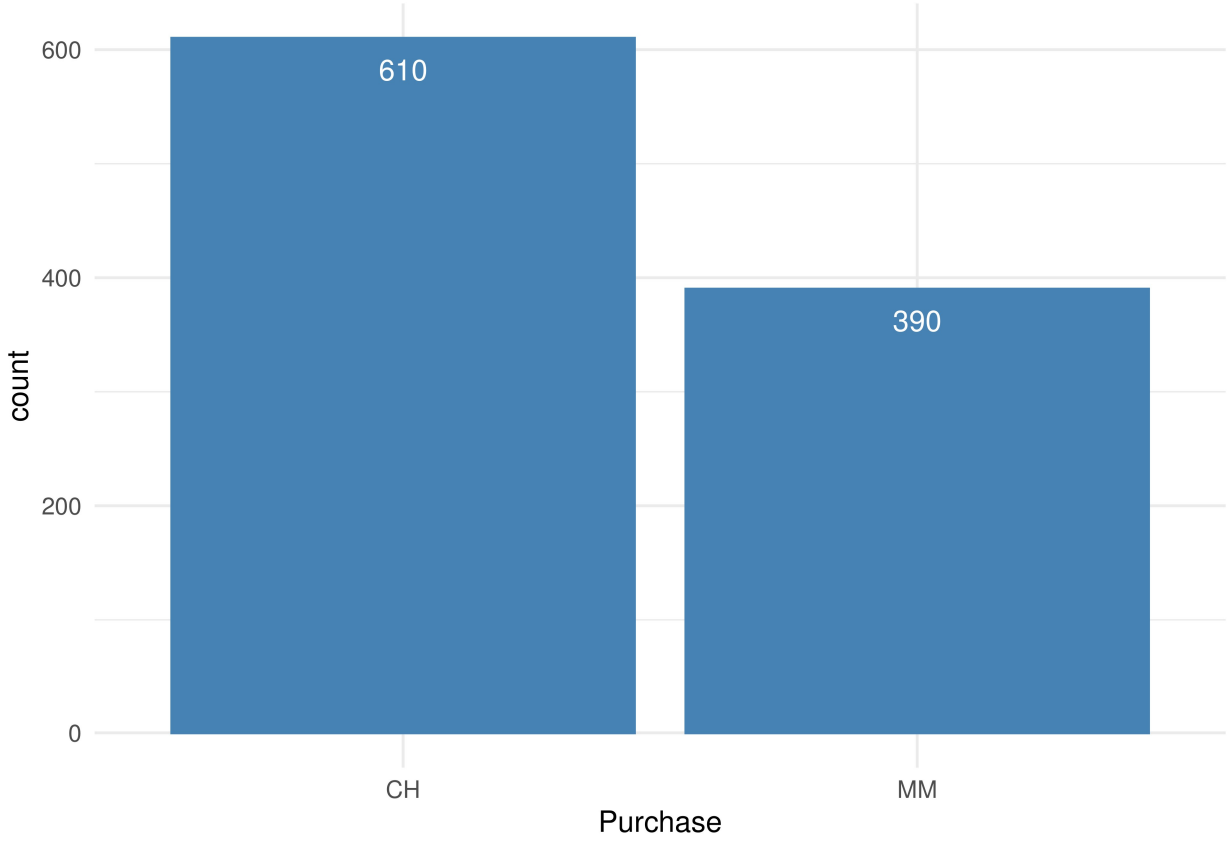
```
dim(juice)
```

```
## [1] 1000   18
```

```
str(juice)
```

```
## '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 ...
```

```
ggplot(juice, aes(x = Purchase)) +
  geom_bar(position = "stack", color = 'steelblue', fill="steelblue")+
  theme_minimal() +
  geom_text(stat='count', aes(label=..count..), vjust=2, color = 'white')
```



All variables has no missing values

```
corr_juice <- juice[,-c(1,14)]
head(corr_juice)
```

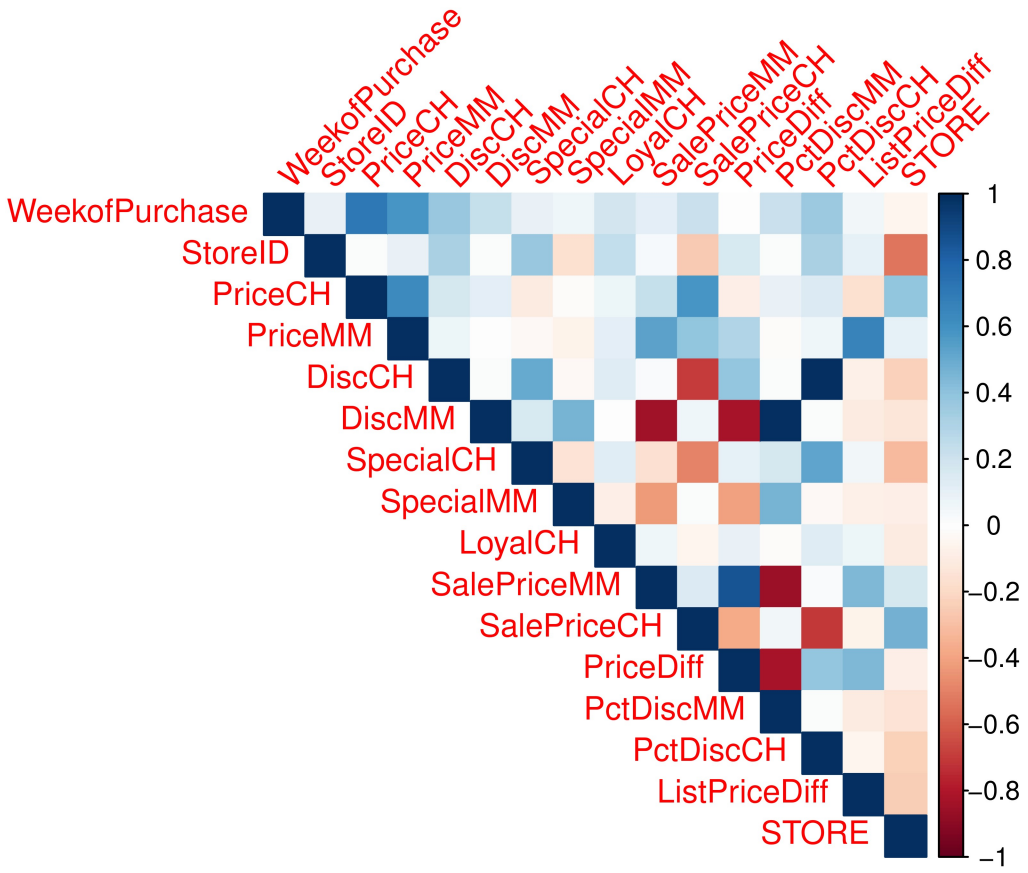
```
##   WeekofPurchase StoreID PriceCH PriceMM DiscCH DiscMM SpecialCH SpecialMM
## 1            237       2    1.75    1.99      0   0.00         0         0
## 2            258       7    1.86    2.18      0   0.00         0         0
## 3            242       3    1.99    2.23      0   0.00         0         0
## 4            271       2    1.86    2.18      0   0.06         0         0
## 5            276       2    1.99    2.18      0   0.00         0         1
## 6            240       1    1.75    1.99      0   0.30         0         1
##   LoyalCH SalePriceMM SalePriceCH PriceDiff PctDiscMM PctDiscCH ListPriceDiff
## 1 0.40000        1.99        1.75      0.24    0.0000         0          0.24
## 2 0.90814        2.18        1.86      0.32    0.0000         0          0.32
## 3 0.00721        2.23        1.99      0.24    0.0000         0          0.24
## 4 0.78839        2.12        1.86      0.26    0.0275         0          0.32
## 5 0.97251        2.18        1.99      0.19    0.0000         0          0.19
## 6 0.50000        1.69        1.75     -0.06    0.1508         0          0.24
##   STORE
## 1     2
## 2     0
## 3     3
## 4     2
## 5     2
## 6     1
```

```
cor(corr_juice)
```

```
##                 WeekofPurchase StoreID PriceCH  PriceMM  DiscCH   DiscMM
## WeekofPurchase         1.00000  0.0904  0.7143  0.58339  0.3746  0.23744
## StoreID                0.09036  1.0000  0.0146  0.09194  0.3210  0.01443
## PriceCH                0.71431  0.0146  1.0000  0.62382  0.1707  0.11347
## PriceMM                0.58339  0.0919  0.6238  1.00000  0.0732  0.00592
## DiscCH                 0.37464  0.3210  0.1707  0.07315  1.0000  0.01902
## DiscMM                 0.23744  0.0144  0.1135  0.00592  0.0190  1.00000
## SpecialCH              0.09522  0.3703 -0.1046 -0.03280  0.5059  0.16722
## SpecialMM              0.06741 -0.1669 -0.0182 -0.06995 -0.0318  0.46000
## LoyalCH                0.18577  0.2420  0.0716  0.11031  0.1320 -0.00951
## SalePriceMM            0.11236  0.0372  0.2390  0.53219  0.0232 -0.84346
## SalePriceCH            0.21173 -0.2531  0.5871  0.39361 -0.6974  0.06691
## PriceDiff             -0.00524  0.1671 -0.0825  0.29334  0.3860 -0.82565
## PctDiscMM              0.21843  0.0186  0.0965 -0.01411  0.0152  0.99880
## PctDiscCH              0.36410  0.3231  0.1532  0.06833  0.9990  0.01953
## ListPriceDiff          0.05629  0.1021 -0.1646  0.66821 -0.0702 -0.10055
## STORE                 -0.05192 -0.5315  0.3926  0.10422 -0.2280 -0.13647
##                 SpecialCH SpecialMM  LoyalCH SalePriceMM SalePriceCH PriceDiff
## WeekofPurchase     0.0952    0.0674  0.18577      0.1124      0.2117  -0.00524
## StoreID            0.3703   -0.1669  0.24201      0.0372     -0.2531   0.16705
## PriceCH           -0.1046   -0.0182  0.07155      0.2390      0.5871  -0.08251
## PriceMM           -0.0328   -0.0700  0.11031      0.5322      0.3936   0.29334
## DiscCH             0.5059   -0.0318  0.13196      0.0232     -0.6974   0.38599
## DiscMM             0.1672    0.4600 -0.00951     -0.8435      0.0669  -0.82565
## SpecialCH          1.0000   -0.1485  0.12713     -0.1592     -0.4917   0.10758
```

```
## SpecialMM        -0.1485    1.0000 -0.08548     -0.4270      0.0128  -0.40703
## LoyalCH           0.1271   -0.0855  1.00000      0.0673     -0.0564   0.09254
## SalePriceMM      -0.1592   -0.4270  0.06731      1.0000      0.1548   0.85661
## SalePriceCH      -0.4917    0.0128 -0.05638      0.1548      1.0000  -0.37714
## PriceDiff         0.1076   -0.4070  0.09254      0.8566     -0.3771   1.00000
## PctDiscMM         0.1712    0.4612 -0.01171     -0.8532      0.0576  -0.82995
## PctDiscCH         0.5209   -0.0361  0.13131      0.0202     -0.7094   0.38939
## ListPriceDiff     0.0582   -0.0709  0.07110      0.4441     -0.0621   0.44873
## STORE            -0.3185   -0.0886 -0.10406      0.1715      0.4729  -0.08616
##                PctDiscMM PctDiscCH ListPriceDiff   STORE
## WeekofPurchase    0.2184    0.3641        0.0563 -0.0519
## StoreID           0.0186    0.3231        0.1021 -0.5315
## PriceCH           0.0965    0.1532       -0.1646  0.3926
## PriceMM          -0.0141    0.0683        0.6682  0.1042
## DiscCH            0.0152    0.9990       -0.0702 -0.2280
## DiscMM            0.9988    0.0195       -0.1005 -0.1365
## SpecialCH         0.1712    0.5209        0.0582 -0.3185
## SpecialMM         0.4612   -0.0361       -0.0709 -0.0886
## LoyalCH          -0.0117    0.1313        0.0711 -0.1041
## SalePriceMM      -0.8532    0.0202        0.4441  0.1715
## SalePriceCH       0.0576   -0.7094       -0.0621  0.4729
## PriceDiff        -0.8299    0.3894        0.4487 -0.0862
## PctDiscMM         1.0000    0.0158       -0.1096 -0.1459
## PctDiscCH         0.0158    1.0000       -0.0596 -0.2367
## ListPriceDiff    -0.1096   -0.0596        1.0000 -0.2421
## STORE            -0.1459   -0.2367       -0.2421  1.0000
```

```
corrplot(cor(corr_juice), method = "color", type = "upper", tl.srt = 45)
```



Train-Test Split (80% in train dataset and 20% in test dataset)

```
trainindex <- createDataPartition(juice$Purchase, p=0.8, list= FALSE)
juice_train <- juice[trainindex, ]
juice_test <- juice[-trainindex, ]
```

SVM Models

SVM Linear Model Without Tuning Parameter

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

```
##
## Call:
## svm(formula = Purchase ~ ., data = juice_train, cost = 0.01, kernel = "linear")
##
##
## Parameters:
```

```
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  446
##
##  ( 224 222 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
 ## Performance Evaluation ##
train_pred1 <- predict(svm1, juice_train)

    # confusion matrix
conf.matrix <- table(Predicted = train_pred1, Actual = juice_train$Purchase)
conf.matrix
```

```
##          Actual
## Predicted  CH  MM
##        CH 433  81
##        MM  55 231
```

```
    # train accuracy

train_accu <- (sum(diag(conf.matrix))) / sum(conf.matrix)

test_pred1 <- predict(svm1, juice_test)

    # confusion matrix
conf.matrix <- table(Predicted = test_pred1, Actual = juice_test$Purchase)
conf.matrix
```

```
##          Actual
## Predicted  CH  MM
##        CH 108  19
##        MM  14  59
```

```
    # test accuracy

test_accu <- (sum(diag(conf.matrix))) / sum(conf.matrix)
print('Train Error Rate:')
```

```
## [1] "Train Error Rate:"
```

```
print(1 - train_accu)
```

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

```
print('Test Error Rate')
```

```
## [1] "Test Error Rate"
```

```
print(1 - test_accu)
```

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

Summary Explanation: We have SVM-Type for classification of Purchase. The kernel that we are using is linear and cost parameter is 0.01. We have 446 support vectors. There are 224 belong to first type of purchase and 222 belongs to second type of purchase. There only 2 classes in our classiifcation problem which are CH and MM (showed in Levels).

SVM Linear With Tuned Cost Parameter

```
tunesvm1 <- tune(svm, Purchase~., data = juice_train,
     ranges = list(cost = (0.01:10), kernel = 'linear'))

summary(tunesvm1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost kernel
##  1.01 linear
##
## - best performance: 0.174
##
## - Detailed performance results:
##    cost kernel error dispersion
## 1  0.01 linear 0.179     0.0378
## 2  1.01 linear 0.174     0.0375
## 3  2.01 linear 0.175     0.0373
## 4  3.01 linear 0.175     0.0354
## 5  4.01 linear 0.178     0.0348
## 6  5.01 linear 0.179     0.0354
## 7  6.01 linear 0.178     0.0348
## 8  7.01 linear 0.177     0.0343
## 9  8.01 linear 0.179     0.0339
## 10 9.01 linear 0.179     0.0339
```

```
bestsvm1 <- tunesvm1$best.model
 ## Performance Evaluation ##
best_train_pred1 <- predict(bestsvm1, juice_train)

    # confusion matrix
conf.matrix <- table(Predicted = best_train_pred1, Actual = juice_train$Purchase)
conf.matrix
```

```
##          Actual
## Predicted  CH  MM
##        CH 431  75
##        MM  57 237
```

```
    # best train accuracy

best_train_accu <- (sum(diag(conf.matrix))) / sum(conf.matrix)

best_test_pred1 <- predict(bestsvm1, juice_test)

    # confusion matrix
conf.matrix <- table(Predicted = best_test_pred1, Actual = juice_test$Purchase)
conf.matrix
```

```
##          Actual
## Predicted  CH  MM
##        CH 108  21
##        MM  14  57
```

```
    # best test accuracy

best_test_accu <- (sum(diag(conf.matrix))) / sum(conf.matrix)
print("SVM Linear Best Train Error Rate")
```

```
## [1] "SVM Linear Best Train Error Rate"
```

```
print(1 - best_train_accu)
```

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

```
print("SVM Linear Best Test Error Rate")
```

```
## [1] "SVM Linear Best Test Error Rate"
```

```
print(1 - best_test_accu)
```

```
## [1] 0.175
```

SVM Radial

```
svm_rad <- svm(Purchase~., data=juice_train, cost = 0.01, kernel = 'radial')
summary(svm_rad)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = juice_train, cost = 0.01, kernel = "radial")
##
```

```
##
## 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
```

```
 ## Performance Evaluation ##
train_pred2 <- predict(svm_rad, juice_train)

    # confusion matrix
conf.matrix <- table(Predicted = train_pred2, Actual = juice_train$Purchase)
conf.matrix
```

```
##          Actual
## Predicted  CH  MM
##        CH 488 312
##        MM   0   0
```

```
    # Train accuracy

train_accu2 <- (sum(diag(conf.matrix))) / sum(conf.matrix)

test_pred2 <- predict(svm_rad, juice_test)

    # confusion matrix
conf.matrix <- table(Predicted = test_pred2, Actual = juice_test$Purchase)
conf.matrix
```

```
##          Actual
## Predicted  CH  MM
##        CH 122  78
##        MM   0   0
```

```
    # test accuracy

test_accu2 <- (sum(diag(conf.matrix))) / sum(conf.matrix)
print('Train Error Rate:')
```

```
## [1] "Train Error Rate:"
```

```
print(1 - train_accu2)
```

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

```
print('Test Error Rate')
```

```
## [1] "Test Error Rate"
```

```
print(1 - test_accu2)
```

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

SVM Radial With Tuned Cost Parameter

```
tunesvm2 <- tune(svm, Purchase~., data = juice_train,
     ranges = list(cost = (0.01:10), kernel= 'radial'))

summary(tunesvm2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost kernel
##  1.01 radial
##
## - best performance: 0.174
##
## - Detailed performance results:
##    cost kernel error dispersion
## 1  0.01 radial 0.390     0.0642
## 2  1.01 radial 0.174     0.0341
## 3  2.01 radial 0.185     0.0332
## 4  3.01 radial 0.182     0.0383
## 5  4.01 radial 0.182     0.0392
## 6  5.01 radial 0.186     0.0375
## 7  6.01 radial 0.190     0.0327
## 8  7.01 radial 0.192     0.0345
## 9  8.01 radial 0.192     0.0345
## 10 9.01 radial 0.192     0.0345
```

```
bestsvm2 <- tunesvm2$best.model
print(bestsvm2)
```

```
##
## Call:
```

```
## best.tune(method = svm, train.x = Purchase ~ ., data = juice_train,
##     ranges = list(cost = (0.01:10), kernel = "radial"))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1.01
##
## Number of Support Vectors:  380
```

```
#Apply best cost
best_train_pred2 <- predict(bestsvm2, juice_train)

    # confusion matrix
conf.matrix <- table(Predicted = best_train_pred2, Actual = juice_train$Purchase)
conf.matrix
```

```
##          Actual
## Predicted  CH  MM
##        CH 448  84
##        MM  40 228
```

```
    # best train accuracy

best_train_accu2 <- (sum(diag(conf.matrix))) / sum(conf.matrix)

best_test_pred2 <- predict(bestsvm2, juice_test)

    # confusion matrix
conf.matrix <- table(Predicted = best_test_pred2, Actual = juice_test$Purchase)
conf.matrix
```

```
##          Actual
## Predicted  CH  MM
##        CH 113  22
##        MM   9  56
```

```
    # best test accuracy

best_test_accu2 <- (sum(diag(conf.matrix))) / sum(conf.matrix)
print("SVM Radial Best Train Error Rate")
```

```
## [1] "SVM Radial Best Train Error Rate"
```

```
print(1 - best_train_accu2)
```

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

```
print("SVM Radial Best Test Error Rate'")
```

```
## [1] "SVM Radial Best Test Error Rate'"
```

```
print(1 - best_test_accu2)
```

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

There is an improvement in error rate when it gets smaller with better cost parameter for SVM radial

SVM Polynomial

```
svm_poly <- svm(Purchase~., data=juice_train, cost = 0.01, kernel = 'polynomial', degree = 2)
summary(svm_poly)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = juice_train, cost = 0.01, kernel = "polynomial",
##     degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.01
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  629
##
##  ( 312 317 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
 ## Performance Evaluation ##
train_pred3 <- predict(svm_poly, juice_train)

    # confusion matrix
conf.matrix <- table(Predicted = train_pred3, Actual = juice_train$Purchase)
conf.matrix
```

```
##          Actual
## Predicted  CH  MM
##        CH 488 312
##        MM   0   0
```

```
    # train accuracy

train_accu3 <- (sum(diag(conf.matrix))) / sum(conf.matrix)

test_pred3 <- predict(svm_poly, juice_test)

    # confusion matrix
conf.matrix <- table(Predicted = test_pred3, Actual = juice_test$Purchase)
conf.matrix
```

```
##          Actual
## Predicted  CH  MM
##        CH 122  78
##        MM   0   0
```

```
    # test accuracy

test_accu3 <- (sum(diag(conf.matrix))) / sum(conf.matrix)
print('Train Error Rate:')
```

```
## [1] "Train Error Rate:"
```

```
print(1 - train_accu3)
```

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

```
print('Test Error Rate')
```

```
## [1] "Test Error Rate"
```

```
print(1 - test_accu3)
```

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

SVM Polynomial with Tuned Cost Parameter

```
tunesvm3 <- tune(svm, Purchase~., data = juice_train,
     ranges = list(cost = (0.01:10), kernel= 'polynomial', degree =2))

summary(tunesvm3)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost     kernel degree
```

```
##  5.01 polynomial      2
##
## - best performance: 0.186
##
## - Detailed performance results:
##    cost     kernel degree error dispersion
## 1  0.01 polynomial      2 0.390     0.0428
## 2  1.01 polynomial      2 0.202     0.0459
## 3  2.01 polynomial      2 0.195     0.0501
## 4  3.01 polynomial      2 0.192     0.0511
## 5  4.01 polynomial      2 0.194     0.0538
## 6  5.01 polynomial      2 0.186     0.0498
## 7  6.01 polynomial      2 0.188     0.0475
## 8  7.01 polynomial      2 0.188     0.0475
## 9  8.01 polynomial      2 0.194     0.0465
## 10 9.01 polynomial      2 0.196     0.0472
```

```
bestsvm3 <- tunesvm3$best.model
print(bestsvm3)
```

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = juice_train,
##     ranges = list(cost = (0.01:10), kernel = "polynomial", degree = 2))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  5.01
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  373
```

```
#Apply best cost
best_train_pred3 <- predict(bestsvm3, juice_train)

    # confusion matrix
conf.matrix <- table(Predicted = best_train_pred3, Actual = juice_train$Purchase)
conf.matrix
```

```
##          Actual
## Predicted  CH  MM
##        CH 451  95
##        MM  37 217
```

```
    # best train accuracy

best_train_accu3 <- (sum(diag(conf.matrix))) / sum(conf.matrix)

best_test_pred3 <- predict(bestsvm3, juice_test)
```

```
    # confusion matrix
conf.matrix <- table(Predicted = best_test_pred3, Actual = juice_test$Purchase)
conf.matrix
```

```
##           Actual
## Predicted  CH  MM
##        CH 113  25
##        MM   9  53
```

```
    # best test accuracy

best_test_accu3 <- (sum(diag(conf.matrix))) / sum(conf.matrix)
print("SVM Polynomial Best Train Error Rate")
```

```
## [1] "SVM Polynomial Best Train Error Rate"
```

```
print(1 - best_train_accu3)
```

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

```
print("SVM Polinomial Best Test Error Rate'")
```

```
## [1] "SVM Polinomial Best Test Error Rate'"
```

```
print(1 - best_test_accu3)
```

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

From the above models, at best cost value, SVM with Radial kernel has better (lower) error rate for test set. We would choose a better performance models with lower error rate on test set for prediction on juice Purchase.