SVM in R: Predictive Analysis of Customer Loss

Domain - Telecom
Sudipto Mitra
https://data.world/sudmitra/svm-in-r-predictive-analysis-of-customer-churn

Problem Statement

A mobile network provider is facing a business problem as lot of customers are transitioning to other service providers. This is causing significant loss to the business. The company likes to understand the factors which impact the loss of customers.

Dataset used

Telecom_Data.csv

Variables

Variables	Description
Churn	1 if customer cancelled service, 0 if not
AccountWeeks	number of weeks customer has had active account
ContractRenewal	1 if customer recently renewed contract, 0 if not
DataPlan	1 if customer has data plan, 0 if not
DataUsage	gigabytes of monthly data usage
CustServCalls	number of calls into customer service
DayMins	average daytime minutes per month
DayCalls	average number of daytime calls
MonthlyCharge	average monthly bill
OverageFee	largest overage fee in last 12 months
RoamMins	average number of roaming minutes

Requirement

- **Explore the relationship between the variables.**
- **Develop a predictive model to predict which of the customers will churn out of the network.**

Coding in R

```
# SVM classification on Customer churn #

### Loading package - "readr" ###

library (readr)

Cust_DF = read.csv ("Telecom_Data.csv")
```

```
> head (Cust_DF)
  Churn AccountWeeks ContractRenewal DataPlan DataUsage CustServCalls DayMins DayCalls MonthlyCharge OverageFee RoamMins
1     0          128               1        1       2.7             1   265.1      110            89       9.87     10.0
2     0          107               1        1       3.7             1   161.6      123            82       9.78     13.7
3     0          137               1        0       0.0             0   243.4      114            52       6.06     12.2
4     0           84               0        0       0.0             2   299.4       71            57       3.10      6.6
5     0           75               0        0       0.0             3   166.7      113            41       7.42     10.1
6     0          118               0        0       0.0             0   223.4       98            57      11.03      6.3

> colnames (Cust_DF)
 [1] "Churn"           "AccountWeeks"    "ContractRenewal" "DataPlan"        "DataUsage"       "CustServCalls"
 [7] "DayMins"         "DayCalls"        "MonthlyCharge"   "OverageFee"      "RoamMins"
```

```
### Checking the structure of the variables ###
```

```
> str (Cust_DF)
'data.frame':   3333 obs. of  11 variables:
 $ Churn          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ AccountWeeks   : int  128 107 137 84 75 118 121 147 117 141 ...
 $ ContractRenewal: int  1 1 1 0 0 0 1 0 1 0 ...
 $ DataPlan       : int  1 1 0 0 0 0 1 0 0 1 ...
 $ DataUsage      : num  2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...
 $ CustServCalls  : int  1 1 0 2 3 0 3 0 1 0 ...
 $ DayMins        : num  265 162 243 299 167 ...
 $ DayCalls       : int  110 123 114 71 113 98 88 79 97 84 ...
 $ MonthlyCharge  : num  89 82 52 57 41 57 87.3 36 63.9 93.2 ...
 $ OverageFee     : num  9.87 9.78 6.06 3.1 7.42 ...
 $ RoamMins       : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
```

```
### Checking spread of the data ##
```

```
> summary (Cust_DF)
     Churn         AccountWeeks    ContractRenewal     DataPlan        DataUsage       CustServCalls      DayMins
 Min.   :0.0000   Min.   :  1.0   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :  0.0
 1st Qu.:0.0000   1st Qu.: 74.0   1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:143.7
 Median :0.0000   Median :101.0   Median :1.0000   Median :0.0000   Median :0.0000   Median :1.000   Median :179.4
 Mean   :0.1449   Mean   :101.1   Mean   :0.9031   Mean   :0.2766   Mean   :0.8165   Mean   :1.563   Mean   :179.8
 3rd Qu.:0.0000   3rd Qu.:127.0   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.7800   3rd Qu.:2.000   3rd Qu.:216.4
 Max.   :1.0000   Max.   :243.0   Max.   :1.0000   Max.   :1.0000   Max.   :5.4000   Max.   :9.000   Max.   :350.8
    DayCalls     MonthlyCharge      OverageFee       RoamMins
 Min.   :  0.0   Min.   : 14.00   Min.   : 0.00   Min.   : 0.00
 1st Qu.: 87.0   1st Qu.: 45.00   1st Qu.: 8.33   1st Qu.: 8.50
 Median :101.0   Median : 53.50   Median :10.07   Median :10.30
 Mean   :100.4   Mean   : 56.31   Mean   :10.05   Mean   :10.24
 3rd Qu.:114.0   3rd Qu.: 66.20   3rd Qu.:11.77   3rd Qu.:12.10
 Max.   :165.0   Max.   :111.30   Max.   :18.19   Max.   :20.00
```

```
### Loading package - "psych" ###

library (psych)
```

```
> describe (Cust_DF)
                vars    n   mean    sd median trimmed   mad min    max  range  skew kurtosis   se
Churn              1 3333   0.14  0.35   0.00    0.06  0.00   0   1.00   1.00  2.02     2.07 0.01
AccountWeeks       2 3333 101.06 39.82 101.00  100.77 40.03   1 243.00 242.00  0.10    -0.11 0.69
ContractRenewal    3 3333   0.90  0.30   1.00    1.00  0.00   0   1.00   1.00 -2.72     5.42 0.01
DataPlan           4 3333   0.28  0.45   0.00    0.22  0.00   0   1.00   1.00  1.00    -1.00 0.01
DataUsage          5 3333   0.82  1.27   0.00    0.58  0.00   0   5.40   5.40  1.27     0.04 0.02
CustServCalls      6 3333   1.56  1.32   1.00    1.42  1.48   0   9.00   9.00  1.09     1.72 0.02
DayMins            7 3333 179.78 54.47 179.40  179.85 53.82   0 350.80 350.80 -0.03    -0.02 0.94
DayCalls           8 3333 100.44 20.07 101.00  100.57 19.27   0 165.00 165.00 -0.11     0.24 0.35
MonthlyCharge      9 3333  56.31 16.43  53.50   55.22 15.57  14 111.30  97.30  0.59    -0.02 0.28
OverageFee        10 3333  10.05  2.54  10.07   10.05  2.55   0  18.19  18.19 -0.02     0.02 0.04
RoamMins          11 3333  10.24  2.79  10.30   10.28  2.67   0  20.00  20.00 -0.24     0.60 0.05
```

```
### Checking missing values ####

> sum (is.na (Cust_DF))
[1] 0

#### Observation - No missing values in the dataset. ####

### Checking duplicated values ###

> sum (duplicated (Cust_DF))
[1] 0

### Observation - No duplicated values. ###

### Checking the count of Customer categories ###

> table (Cust_DF $Churn)

   0    1 
2850  483 

### Loading package - "plyr" ###

library (plyr)

> count (Cust_DF $Churn)
  x freq
1 0 2850
2 1  483

#### Observation - The above results show that the count of customers churned
#### out is 483. ####
```

```
### Checking correlation of variables ###

### Loading package - "ggcorrplot" ###
```

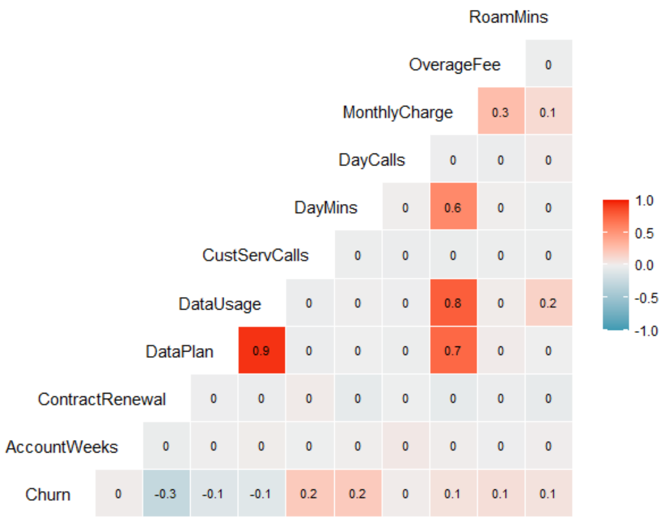
```
> library (ggcorrplot)
Loading required package: ggplot2

Attaching package: 'ggplot2'

The following objects are masked from 'package:psych':

    %+%, alpha

Warning messages:
1: package 'ggcorrplot' was built under R version 4.0.5
2: package 'ggplot2' was built under R version 4.0.4
```

```
> corr = round (cor (Cust_DF), 1)
> head (corr)
                Churn AccountWeeks ContractRenewal DataPlan DataUsage CustServCalls DayMins DayCalls MonthlyCharge
Churn             1.0            0            -0.3     -0.1      -0.1           0.2     0.2        0           0.1
AccountWeeks      0.0            1             0.0      0.0       0.0           0.0     0.0        0           0.0
ContractRenewal  -0.3            0             1.0      0.0       0.0           0.0     0.0        0           0.0
DataPlan         -0.1            0             0.0      1.0       0.9           0.0     0.0        0           0.7
DataUsage        -0.1            0             0.0      0.9       1.0           0.0     0.0        0           0.8
CustServCalls     0.2            0             0.0      0.0       0.0           1.0     0.0        0           0.0
                OverageFee RoamMins
Churn                  0.1      0.1
AccountWeeks           0.0      0.0
ContractRenewal        0.0      0.0
DataPlan               0.0      0.0
DataUsage              0.0      0.2
CustServCalls          0.0      0.0
```

```
### Loading package - "GGally" ###
```

```
> library (GGally)
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

```
ggcorr (Cust_DF, label = TRUE, label_size = 2.9, hjust = 1, layout.exp = 2)
```



```
#### Observation - High correlations are observed between DataPlan & DataUsage,
#### DataPlan & MonthlyCharge, DataUsage & MonthlyCharge. There's moderate
#### correlation between DayMins & MonthlyCharge. ###

### Changing the data type of the response variable ###
```

```
> class (Cust_DF $Churn)
[1] "integer"
```

```
> Cust_DF $Churn = as.factor (Cust_DF $Churn)   #### Cust_DF $Churn = sapply (Cust_DF $Churn, factor) ####
>
> class (Cust_DF $Churn)
[1] "factor"
>
> str (Cust_DF)
'data.frame':   3333 obs. of  11 variables:
 $ Churn          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ AccountWeeks   : int  128 107 137 84 75 118 121 147 117 141 ...
 $ ContractRenewal: int  1 1 1 0 0 0 1 0 1 0 ...
 $ DataPlan       : int  1 1 0 0 0 0 1 0 0 1 ...
 $ DataUsage      : num  2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...
 $ CustServCalls  : int  1 1 0 2 3 0 3 0 1 0 ...
 $ DayMins        : num  265 162 243 299 167 ...
 $ DayCalls       : int  110 123 114 71 113 98 88 79 97 84 ...
 $ MonthlyCharge  : num  89 82 52 57 41 57 87.3 36 63.9 93.2 ...
 $ OverageFee     : num  9.87 9.78 6.06 3.1 7.42 ...
 $ RoamMins       : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
```

```
## Splitting observational data into 70:30 ratio for training/testing model ##
```

```
> Sampl_sz = floor (0.7 * nrow (Cust_DF))
> Sampl_sz
[1] 2333

> set.seed (123)
>
> train_ind = sample (seq_len (nrow (Cust_DF)), size = Sampl_sz)
>
> train_data = Cust_DF [train_ind,]
>
> test_data = Cust_DF [- train_ind,]
>
> dim (train_data)
[1] 2333   11
>
> dim (test_data)
[1] 1000   11
```

```
## Developing SVM Model ##

### Loading/Installing package - "e1071" ###

library (e1071)
```

```
> svm.model = svm (Churn ~ ., data = train_data)
>
> svm.model

Call:
svm(formula = Churn ~ ., data = train_data)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  663
```

```
> summary (svm.model)

Call:
svm(formula = Churn ~ ., data = train_data)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  663

 ( 369 294 )


Number of Classes:  2

Levels:
 0 1
```

```
## Predicting values ##

> preds = predict (svm.model, test_data)
>
> tab = table (preds, test_data $Churn)
>
> tab

preds   0   1
    0 849  75
    1   8  68
```

```
## Computing classification Accuracy ##
## Loading package - "caret" ##

> library (caret)
Loading required package: lattice
Warning message:
package 'caret' was built under R version 4.0.4
```

```
> confusionMatrix (table (preds, test_data $Churn))
Confusion Matrix and Statistics


preds   0   1
    0 849  75
    1   8  68

               Accuracy : 0.917
                 95% CI : (0.8981, 0.9334)
    No Information Rate : 0.857
    P-Value [Acc > NIR] : 4.482e-09

                  Kappa : 0.5792

 Mcnemar's Test P-Value : 4.342e-13

            Sensitivity : 0.9907
            Specificity : 0.4755
         Pos Pred Value : 0.9188
         Neg Pred Value : 0.8947
             Prevalence : 0.8570
         Detection Rate : 0.8490
   Detection Prevalence : 0.9240
      Balanced Accuracy : 0.7331

       'Positive' Class : 0
```

```
### Accuracy observed as 91.7%. ###
```

```
> ## Computing Misclassification Error Rate ##
>
> 1 - (sum (diag (tab)) / sum (tab))
[1] 0.083
```

```
### Observation - Misclassification Error rate computed as 8.3%. ###
```

```
## Developing SVM model with Kernel parameter as "linear" ##
```

```
> svm_model = svm (Churn ~ ., data = train_data, kernel = "linear")
>
> svm_model

Call:
svm(formula = Churn ~ ., data = train_data, kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  879

>
> summary (svm_model)

Call:
svm(formula = Churn ~ ., data = train_data, kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  879

 ( 539 340 )


Number of Classes:  2

Levels:
 0 1
```

```
### Observation - No. of support vectors increased. ###

## Predicting values ##

> preds1 = predict (svm_model, test_data)
>
> tab1 = table (preds1, test_data $Churn)

## Computing classification accuracy ##
## Loading package - "caret" ##

> confusionMatrix (table (preds1, test_data $Churn))
Confusion Matrix and Statistics


preds1   0   1
     0 857 143
     1   0   0

               Accuracy : 0.857
                 95% CI : (0.8338, 0.8781)
    No Information Rate : 0.857
    P-Value [Acc > NIR] : 0.5223

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.000
            Specificity : 0.000
         Pos Pred Value : 0.857
         Neg Pred Value :   NaN
             Prevalence : 0.857
         Detection Rate : 0.857
   Detection Prevalence : 1.000
      Balanced Accuracy : 0.500

       'Positive' Class : 0

### Observation - Accuracy decreased to 85.7% ###

## Computing Misclassification Error Rate ##

> 1 - (sum (diag (tab1)) / sum (tab1))
[1] 0.143

### Observation - Error rate increased to 14.3%. ###
```

```
## Developing SVM model with kernel parameter as "polynomial" ##

> svm.model.poly = svm (Churn ~ ., data = train_data, kernel = "polynomial")
>
> svm.model.poly

Call:
svm(formula = Churn ~ ., data = train_data, kernel = "polynomial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  1
     degree:  3
     coef.0:  0

Number of Support Vectors:  558

> summary (svm.model.poly)

Call:
svm(formula = Churn ~ ., data = train_data, kernel = "polynomial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  1
     degree:  3
     coef.0:  0

Number of Support Vectors:  558

 ( 296 262 )


Number of Classes:  2

Levels:
 0 1


## Predicting on test data ##

> preds_poly = predict (svm.model.poly, test_data)
>
> tab_poly = table (preds_poly, test_data $Churn)
```

```
## Computing classification Accuracy ##
## Loading package - "caret" ##
```

```
> confusionMatrix (table (preds_poly, test_data $Churn))
Confusion Matrix and Statistics


preds_poly   0   1
         0 846  64
         1  11  79

               Accuracy : 0.925
                 95% CI : (0.9069, 0.9406)
    No Information Rate : 0.857
    P-Value [Acc > NIR] : 2.041e-11

                  Kappa : 0.6381

 Mcnemar's Test P-Value : 1.920e-09

            Sensitivity : 0.9872
            Specificity : 0.5524
         Pos Pred Value : 0.9297
         Neg Pred Value : 0.8778
             Prevalence : 0.8570
         Detection Rate : 0.8460
   Detection Prevalence : 0.9100
      Balanced Accuracy : 0.7698

       'Positive' Class : 0
```

```
### Observation - Accuracy increased to 92.5%. ###
```

```
## Computing Misclassification Error Rate ##
```

```
> 1 - (sum (diag (tab_poly)) / sum (tab_poly))
[1] 0.075
```

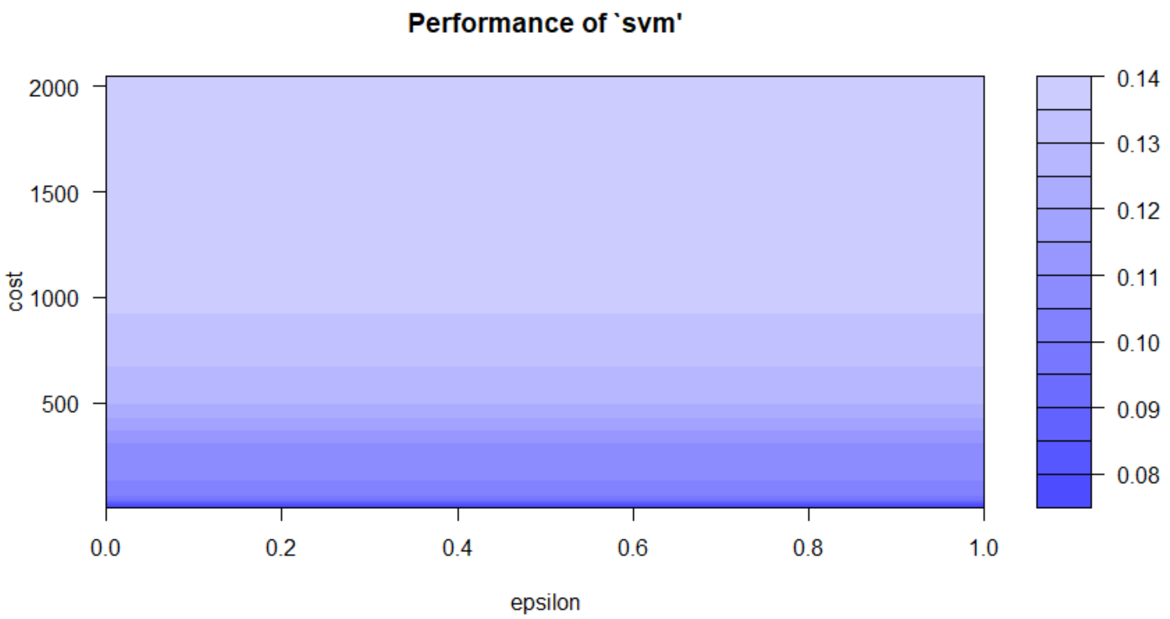
```
### Observation - Error rate decreased to 7.5%. ###
```

```
## Developing SVM model with kernel parameter as "sigmoid" ##
```

```
> svm.model.sig = svm (Churn ~ ., data = train_data, kernel = "sigmoid")
>
> svm.model.sig

Call:
svm(formula = Churn ~ ., data = train_data, kernel = "sigmoid")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  sigmoid
       cost:  1
     coef.0:  0

Number of Support Vectors:  631
```

```
> summary (svm.model.sig)

Call:
svm(formula = Churn ~ ., data = train_data, kernel = "sigmoid")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  sigmoid
       cost:  1
     coef.0:  0

Number of Support Vectors:  631

 ( 317 314 )


Number of Classes:  2

Levels:
 0 1

## Predicting values ##

> preds_sig = predict (svm.model.sig, test_data)
>
> tab_sig = table (preds_sig, test_data $Churn)

## Computing classification accuracy ##
## Loading package - "caret" ##

> confusionMatrix (table (preds_sig, test_data $Churn))
Confusion Matrix and Statistics


preds_sig   0   1
        0 779 125
        1  78  18

               Accuracy : 0.797
                 95% CI : (0.7707, 0.8215)
    No Information Rate : 0.857
    P-Value [Acc > NIR] : 1.000000

                  Kappa : 0.0404

 Mcnemar's Test P-Value : 0.001244

            Sensitivity : 0.9090
            Specificity : 0.1259
         Pos Pred Value : 0.8617
         Neg Pred Value : 0.1875
             Prevalence : 0.8570
         Detection Rate : 0.7790
   Detection Prevalence : 0.9040
      Balanced Accuracy : 0.5174

       'Positive' Class : 0

### Observation - Accuracy rate is highly dipped to 79.7%. ###
```

```
## Computing Misclassification Error Rate ##

> 1 - (sum (diag (tab_sig)) / sum (tab_sig))
[1] 0.203

### Error rate is highly increased to 20.3%. ###

### Observation - The SVM model with kernel parameter as "polynomial" has
### the least misclassification error rate & highest accuracy. ###

## Tuning of model ##

set.seed (123)

tmodel = tune (svm, Churn ~ ., data = test_data,
               ranges = list (epsilon = seq (0, 1, 0.1), cost = 2 ^ (2 : 11)))

## Plotting the model tuned ##

plot (tmodel)
```

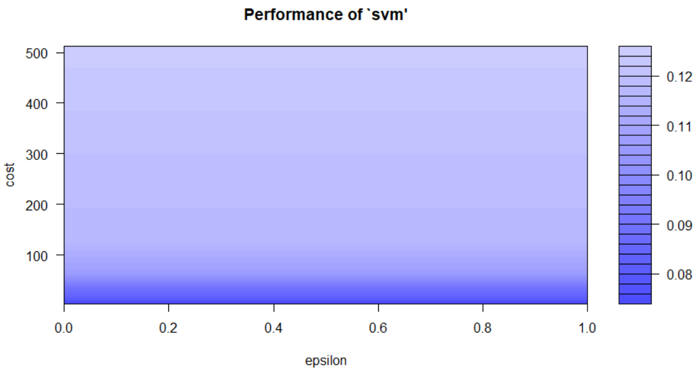



```
### Observation - The plot returns performance evaluation of SVM for the
### 2 parameters (cost, epsilon) used. The darker regions towards the bottom
### indicate lower misclassification error. It also suggests that if we
### confine our search till 512 instead of 2048, it will probably make model
### more accurate. ###
```

```
tmodel1 = tune (svm, Churn ~ ., data = test_data,
                ranges = list (epsilon = seq (0, 1, 0.1), cost = 2 ^ (2 : 9)))
```

```
## Plotting the model re-tuned ##
```

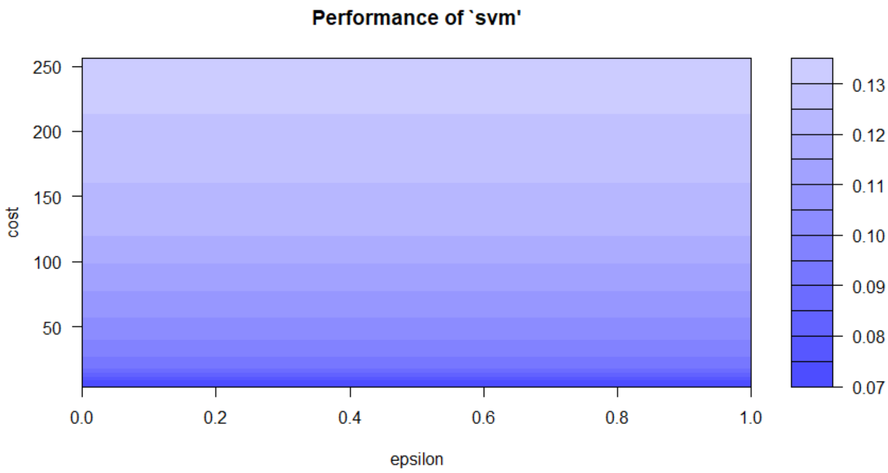
```
plot (tmodel1)
```




```
### Observation - The plot returns performance evaluation of SVM for the
### 2 parameters (cost, epsilon) used. The darker regions in the bottom
### indicate lower misclassification error. It also suggests that if we
### confine our search till 200 instead of 512, it will probably make model
### more accurate. ###
```

```
tmodel2 = tune (svm, Churn ~ ., data = test_data,
                ranges = list (epsilon = seq (0, 1, 0.1), cost = 2 ^ (2 : 8)))
```

```
## Plotting the model re-tuned ##

plot (tmodel2)
```




```
### Observation - The darker regions in the bottom indicate lower
### misclassification error. ###
```

```
> summary (tmodel2)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 epsilon cost
       0    4

- best performance: 0.074

- Detailed performance results:
   epsilon cost error dispersion
1      0.0    4 0.074 0.02547330
2      0.1    4 0.074 0.02547330
3      0.2    4 0.074 0.02547330
4      0.3    4 0.074 0.02547330
5      0.4    4 0.074 0.02547330
6      0.5    4 0.074 0.02547330
7      0.6    4 0.074 0.02547330
8      0.7    4 0.074 0.02547330
9      0.8    4 0.074 0.02547330
10     0.9    4 0.074 0.02547330
11     1.0    4 0.074 0.02547330
```

```
### Observation - The summary result shows best parameters as epsilon - 0 &
### cost - 4, i.e., 2^2. The best SVM classification model can be chosen
### using this summary result. ###
```

```
## Best SVM Model ##

Best.svm = tmodel2 $best.model
```

```
> summary (Best.svm)

Call:
best.tune(method = svm, train.x = Churn ~ ., data = test_data, ranges = list(epsilon = seq(0, 1,
    0.1), cost = 2^(2:8)))


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  4

Number of Support Vectors:  279

 ( 171 108 )


Number of Classes:  2

Levels:
 0 1
```

```
### Observation - The summary shows 279 nos. of support vectors with 171, 108,
### nos. of support vectors for each of the 2 classes of the response
### variable. It also shows kernel parameter as "radial"
### & the cost value as 4. ###
```

```
## Predicting on test data ##

preds_best = predict (Best.svm, test_data)

tab_Best = table (preds_best, test_data $Churn)
```

```
## Computing SVM Classification Accuracy ##
## Loading package - "caret" ##
```

```
> confusionMatrix (table (preds_best, test_data $Churn))
Confusion Matrix and Statistics


preds_best   0   1
         0 854  35
         1   3 108

               Accuracy : 0.962
                 95% CI : (0.9482, 0.973)
    No Information Rate : 0.857
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.829

 Mcnemar's Test P-Value : 4.934e-07

            Sensitivity : 0.9965
            Specificity : 0.7552
         Pos Pred Value : 0.9606
         Neg Pred Value : 0.9730
             Prevalence : 0.8570
         Detection Rate : 0.8540
   Detection Prevalence : 0.8890
      Balanced Accuracy : 0.8759

       'Positive' Class : 0
```

```
### Observation - SVM model Accuracy sharply increased to 96.2%. ###
```

```
## Computing Misclassification Error Rate ##
```

```
> 1 - (sum (diag (tab_Best)) / sum (tab_Best))
[1] 0.038
```

```
### Misclassification Error rate has remarkably decreased to 3.8%. ###
```