

SVM - CARET PACKAGE

Sruthi Kizhakathra

July 6, 2018

I am implementing SVM using Caret library in R. Hence installing Caret and all other libraries required for data pre processing and wrangling.

```
library(readxl)
library(gower)
```

```
## Warning: package 'gower' was built under R version 3.4.4
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.4.4
```

Loading the source data from the csv file.

```
heart_df <- read.csv("C:/Users/Sruthi/Desktop/heart_tidy.csv", sep = ',', header = FALSE)
```

Next we will do some initial analysis of the dataset.

Heart Disease data set consists of 14 numeric attributes with 300 rows of data. 14th variable, V14 is the response variable/label and rest 13 variables, V1 to V13 will be used in our prediction problem.

```
str(heart_df)
```

```
## 'data.frame':   300 obs. of  14 variables:
## $ V1 : int  63 67 67 37 41 56 62 57 63 53 ...
## $ V2 : int  1 1 1 1 0 1 0 0 1 1 ...
## $ V3 : int  1 4 4 3 2 2 4 4 4 4 ...
## $ V4 : int  145 160 120 130 130 120 140 120 130 140 ...
## $ V5 : int  233 286 229 250 204 236 268 354 254 203 ...
## $ V6 : int  1 0 0 0 0 0 0 0 0 1 ...
## $ V7 : int  2 2 2 0 2 0 2 0 2 2 ...
## $ V8 : int  150 108 129 187 172 178 160 163 147 155 ...
## $ V9 : int  0 1 1 0 0 0 0 1 0 1 ...
## $ V10: num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
## $ V11: int  3 2 2 3 1 1 3 1 2 3 ...
## $ V12: int  0 3 2 0 0 0 2 0 1 0 ...
## $ V13: int  6 3 7 3 3 3 3 3 7 7 ...
## $ V14: int  0 1 1 0 0 0 1 0 1 1 ...
```

Further inspection shows that there are no missing values within the data which makes our process easier.

```
summary(heart_df);any(NA)
```

```
##           V1           V2           V3           V4
## Min.      :29.00   Min.      :0.00   Min.      :1.000   Min.      : 94.0
## 1st Qu.:48.00   1st Qu.:0.00   1st Qu.:3.000   1st Qu.:120.0
```

```
## Median :56.00 Median :1.00 Median :3.000 Median :130.0
## Mean :54.48 Mean :0.68 Mean :3.153 Mean :131.6
## 3rd Qu.:61.00 3rd Qu.:1.00 3rd Qu.:4.000 3rd Qu.:140.0
## Max. :77.00 Max. :1.00 Max. :4.000 Max. :200.0
## V5 V6 V7 V8
## Min. :126.0 Min. :0.0000 Min. :0.0000 Min. : 71.0
## 1st Qu.:211.0 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:133.8
## Median :241.5 Median :0.0000 Median :0.5000 Median :153.0
## Mean :246.9 Mean :0.1467 Mean :0.9867 Mean :149.7
## 3rd Qu.:275.2 3rd Qu.:0.0000 3rd Qu.:2.0000 3rd Qu.:166.0
## Max. :564.0 Max. :1.0000 Max. :2.0000 Max. :202.0
## V9 V10 V11 V12
## Min. :0.0000 Min. :0.00 Min. :1.000 Min. :0.00
## 1st Qu.:0.0000 1st Qu.:0.00 1st Qu.:1.000 1st Qu.:0.00
## Median :0.0000 Median :0.80 Median :2.000 Median :0.00
## Mean :0.3267 Mean :1.05 Mean :1.603 Mean :0.67
## 3rd Qu.:1.0000 3rd Qu.:1.60 3rd Qu.:2.000 3rd Qu.:1.00
## Max. :1.0000 Max. :6.20 Max. :3.000 Max. :3.00
## V13 V14
## Min. :3.000 Min. :0.00
## 1st Qu.:3.000 1st Qu.:0.00
## Median :3.000 Median :0.00
## Mean :4.727 Mean :0.46
## 3rd Qu.:7.000 3rd Qu.:1.00
## Max. :7.000 Max. :1.00

## [1] NA
```

Since the target variable is integer, we will convert this into factor variable

```
heart_df$V14 <- as.factor(heart_df$V14)
str(heart_df)
```

```
## 'data.frame': 300 obs. of 14 variables:
## $ V1 : int 63 67 67 37 41 56 62 57 63 53 ...
## $ V2 : int 1 1 1 1 0 1 0 0 1 1 ...
## $ V3 : int 1 4 4 3 2 2 4 4 4 4 ...
## $ V4 : int 145 160 120 130 130 120 140 120 130 140 ...
## $ V5 : int 233 286 229 250 204 236 268 354 254 203 ...
## $ V6 : int 1 0 0 0 0 0 0 0 0 1 ...
## $ V7 : int 2 2 2 0 2 0 2 0 2 2 ...
## $ V8 : int 150 108 129 187 172 178 160 163 147 155 ...
## $ V9 : int 0 1 1 0 0 0 0 1 0 1 ...
## $ V10: num 2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
## $ V11: int 3 2 2 3 1 1 3 1 2 3 ...
## $ V12: int 0 3 2 0 0 0 2 0 1 0 ...
## $ V13: int 6 3 7 3 3 3 3 3 7 7 ...
## $ V14: Factor w/ 2 levels "0","1": 1 2 2 1 1 1 2 1 2 2 ...
```

Before proceeding to modelling part, lets do test train split of our dataset. The training data set will be used in our modelling process where as the test data set will be used to validate/evaluste our model. We will use 70/30 split for train/test data set.

specifying y variable ensures that our data is being partitioned on the target variable V14.

```
set.seed(3033)
intrain <- createDataPartition(y = heart_df$V14, p= 0.7, list = FALSE)
```

```
training <- heart_df[intrain,]
testing <- heart_df[-intrain,]
```

Lets look into train/test data sets

```
dim(training);dim(testing)
```

```
## [1] 211 14
```

```
## [1] 89 14
```

```
training[["V14"]] = factor(training[["V14"]])
```

Modelling

Firstly, we will define the trainControl for our model. In this step, we will define 3 parameters:

- (1) method: type of resampling. Here we will use “repeatedcv”
- (2) number: no of resampling iterations. We will specify 10.
- (3) repeats: Complete sets of folds for our repeated cross-validation. Here I am using 3.

```
set.seed(3233)
```

```
trnctrl <- trainControl(method = "repeatedcv",number=10,repeats=3)
```

While training SVM, we are specifying some additional variables: [1] method: training algorithm [2] preprocess : centre and scale will convert training data with mean 0 and sd 1 after preprocessing. [3] tuneLength: For tuning our algorithm and holds a integer value. [4] metric : Model evaluation metric . The different options available are Accuracy and Kappa RMSE and R² ROC (AUC, Sensitivity and Specificity) LogLoss.

I will be using Accuracy here as it is a classification problem. If we are going with a regression problem then we can go for metrics like RMSE.

Kappa coefficient is good for evaluating imbalanced class classifier. It is like classification accuracy, except that it is normalized at the baseline of random chance on your dataset.

```
svmLinear <- train(V14~.,data=training, method="svmLinear", metric=c("Accuracy"),
  trControl=trnctrl,
  preProcess=c("center","scale"),
  tuneLength=10)
```

Next lets look into our model results from the svmLinear variable. We can see that the accuracy of the model is 81.2% and the Kappa the other hand shows approximately 62% which is interesting.

```
svmLinear
```

```
## Support Vector Machines with Linear Kernel
##
## 211 samples
## 13 predictor
## 2 classes: '0', '1'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 190, 189, 190, 190, 190, 191, ...
## Resampling results:
##
## Accuracy Kappa
```

```
## 0.8375685 0.6714301
```

```
##
```

```
## Tuning parameter 'C' was held constant at a value of 1
```

Since this is a binary class problem, lets look into ROC curve metric as well.

```
levels(training$V14) <- make.names(levels(factor(training$V14))) #class values must be factors and must
```

```
control <- trainControl(method="repeatedcv", number=10, classProbs=TRUE, summaryFunction=twoClassSummary,
set.seed(7)
```

```
svmLinear_ROC <- train(V14~.,data=training, method="svmLinear", metric="ROC",
trControl=control
)
```

Lets look into the ROC curve as well. The model is a pretty decent model with area under ROC curve as .878, which is a fairly high value.

```
svmLinear_ROC
```

```
## Support Vector Machines with Linear Kernel
```

```
##
```

```
## 211 samples
```

```
## 13 predictor
```

```
## 2 classes: 'X0', 'X1'
```

```
##
```

```
## No pre-processing
```

```
## Resampling: Cross-Validated (10 fold, repeated 1 times)
```

```
## Summary of sample sizes: 189, 191, 189, 190, 191, 190, ...
```

```
## Resampling results:
```

```
##
```

```
## ROC Sens Spec
```

```
## 0.9026178 0.8674242 0.7955556
```

```
##
```

```
## Tuning parameter 'C' was held constant at a value of 1
```

Now we know that our model performs good with training data , lets use test set to do our predictions.

```
test <- predict(svmLinear,testing)
```

```
test
```

```
## [1] 1 1 1 1 1 0 0 0 0 1 0 0 1 0 0 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0 0 0 0
```

```
## [36] 0 1 0 1 0 0 0 1 0 0 1 1 0 0 1 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0
```

```
## [71] 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1
```

```
## Levels: 0 1
```

Inorder to see how accurately model is performing on test set, lets look into confusion matrix of test set. The model shows an accuracy of 77.5% in test dataset.

```
confusionMatrix(test,testing$V14)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
## Reference
```

```
## Prediction 0 1
```

```
## 0 39 11
```

```
## 1 9 30
```

```
##
```

```
## Accuracy : 0.7753
```

```
##          95% CI : (0.6745, 0.857)
##      No Information Rate : 0.5393
##      P-Value [Acc > NIR] : 3.423e-06
##
##          Kappa : 0.5461
##  McNemar's Test P-Value : 0.8231
##
##      Sensitivity : 0.8125
##      Specificity : 0.7317
##      Pos Pred Value : 0.7800
##      Neg Pred Value : 0.7692
##      Prevalence : 0.5393
##      Detection Rate : 0.4382
##      Detection Prevalence : 0.5618
##      Balanced Accuracy : 0.7721
##
##      'Positive' Class : 0
##
```

Lets tune our classifier by including C parameter or cost parameter

```
grid <- expand.grid(C=c(0.0,0.1,0.2,.3,.4,.5,.6,.7,.8,.9,1.0))
set.seed(3233)
svm_Linear_Grid <- train(V14 ~., data = training, method = "svmLinear", metric="Accuracy",
                        trControl=control,
                        preProcess = c("center", "scale"),
                        tuneGrid = grid,
                        tuneLength = 10)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was
## not in the result set. ROC will be used instead.
##
## Warning: model fit failed for Fold01.Rep1: C=0.0 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
##
## Warning: model fit failed for Fold02.Rep1: C=0.0 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
##
## Warning: model fit failed for Fold03.Rep1: C=0.0 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
##
## Warning: model fit failed for Fold04.Rep1: C=0.0 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
##
## Warning: model fit failed for Fold05.Rep1: C=0.0 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
##
## Warning: model fit failed for Fold06.Rep1: C=0.0 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
##
## Warning: model fit failed for Fold07.Rep1: C=0.0 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
##
## Warning: model fit failed for Fold08.Rep1: C=0.0 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
##
## Warning: model fit failed for Fold09.Rep1: C=0.0 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
##
## Warning: model fit failed for Fold10.Rep1: C=0.0 Error in .local(x, ...) :
```

```
## No Support Vectors found. You may want to change your parameters
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
svm_Linear_Grid
```

```
## Support Vector Machines with Linear Kernel
##
## 211 samples
## 13 predictor
## 2 classes: 'X0', 'X1'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 190, 189, 190, 190, 190, 191, ...
## Resampling results across tuning parameters:
##
## C ROC Sens Spec
## 0.0 0.4407239 NaN NaN
## 0.1 0.8914226 0.8689394 0.8033333
## 0.2 0.8903367 0.8689394 0.8033333
## 0.3 0.8938047 0.8598485 0.8133333
## 0.4 0.8938215 0.8507576 0.8133333
## 0.5 0.8938047 0.8598485 0.8033333
## 0.6 0.8931481 0.8507576 0.8133333
## 0.7 0.8922391 0.8507576 0.8033333
## 0.8 0.8903872 0.8598485 0.7933333
## 0.9 0.8903872 0.8507576 0.8033333
## 1.0 0.8884680 0.8689394 0.7933333
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.4.
```

The model accuracy improved from 77 to 80.9%.

```
test_grid <-predict(svm_Linear_Grid,testing)
levels(testing$V14) <- make.names(levels(factor(testing$V14)))
levels(test_grid) <- make.names(levels(factor(test_grid)))
test_grid
```

```
## [1] X0 X1 X1 X1 X1 X0 X0 X0 X0 X1 X0 X0 X1 X0 X0 X1 X1 X1 X0 X0 X1 X0 X1
## [24] X0 X1 X1 X1 X1 X1 X0 X1 X0 X0 X0 X0 X0 X0 X0 X1 X0 X0 X0 X1 X0 X0 X1
## [47] X1 X0 X0 X1 X1 X1 X1 X0 X0 X1 X0 X1 X1 X1 X0 X1 X0 X0 X1 X0 X0 X0 X0
## [70] X0 X0 X0 X0 X0 X1 X0 X0 X1 X1 X0 X0 X0 X0 X0 X0 X0 X1 X0 X1
## Levels: X0 X1
```

```
confusionMatrix(test_grid,testing$V14)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction X0 X1
##           X0 42 11
##           X1  6 30
##
##           Accuracy : 0.809
```

```
##          95% CI : (0.7119, 0.8846)
##      No Information Rate : 0.5393
##      P-Value [Acc > NIR] : 9.657e-08
##
##          Kappa : 0.6122
##  McNemar's Test P-Value : 0.332
##
##      Sensitivity : 0.8750
##      Specificity : 0.7317
##      Pos Pred Value : 0.7925
##      Neg Pred Value : 0.8333
##      Prevalence : 0.5393
##      Detection Rate : 0.4719
##      Detection Prevalence : 0.5955
##      Balanced Accuracy : 0.8034
##
##      'Positive' Class : X0
##
```

SVM Classifier using Non-Linear Kernel

Next, we will try to build a model using Non-Linear Kernel with Radial Basis Function. Here we need to select proper value of Cost “C” parameter and “sigma” parameter.

The model exhibits higher performance of 84% accuracy with C=0.25 and sigma =0.05492403

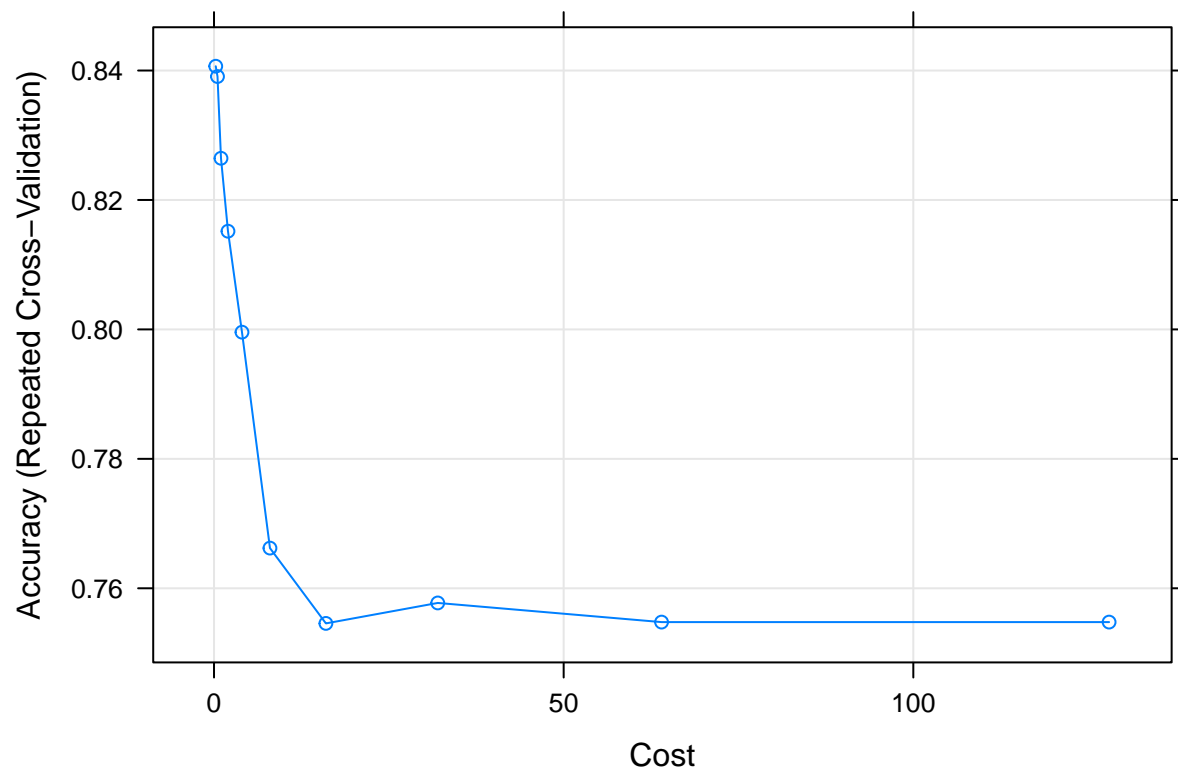
```
set.seed(3233)
svm_Radial <- train(V14 ~., data = training, method = "svmRadial",
  trControl=trnctrl,
  preProcess = c("center", "scale"),
  tuneLength = 10)

svm_Radial
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 211 samples
## 13 predictor
## 2 classes: 'X0', 'X1'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 190, 189, 190, 190, 190, 191, ...
## Resampling results across tuning parameters:
##
##      C          Accuracy      Kappa
##      0.25  0.8406638  0.6776913
##      0.50  0.8390765  0.6746159
##      1.00  0.8264430  0.6490847
##      2.00  0.8151732  0.6265426
##      4.00  0.7995743  0.5953991
##      8.00  0.7662121  0.5276444
##     16.00  0.7545743  0.5058932
##     32.00  0.7577345  0.5138661
##     64.00  0.7547763  0.5074463
##    128.00  0.7547763  0.5074463
```

```
##
## Tuning parameter 'sigma' was held constant at a value of 0.05492403
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.05492403 and C = 0.25.
```

```
plot(svm_Radial)
```



We are getting an accuracy of 87.78%.

```
test_radial <- predict(svm_Radial,testing)
test_radial
```

```
## [1] X1 X1 X1 X1 X1 X0 X0 X0 X0 X0 X1 X0 X0 X1 X0 X0 X1 X1 X1 X0 X0 X1 X0 X0
## [24] X0 X1 X0 X1 X1 X1 X0 X1 X0 X0 X0 X0 X0 X0 X0 X1 X0 X0 X0 X1 X0 X0 X1
## [47] X1 X0 X0 X1 X1 X1 X1 X0 X0 X1 X0 X1 X1 X1 X0 X0 X0 X0 X1 X0 X0 X0 X0
## [70] X0 X0 X1 X0 X0 X1 X0 X0 X1 X1 X0 X0 X0 X0 X0 X0 X0 X0 X0 X0 X1
## Levels: X0 X1
```

```
levels(testing$V14)
```

```
## [1] "X0" "X1"
```

```
confusionMatrix(test_radial,testing$V14)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction X0 X1
##          X0 44 11
```



```
##           X1  4 30
##
##           Accuracy : 0.8315
##           95% CI : (0.7373, 0.9025)
##           No Information Rate : 0.5393
##           P-Value [Acc > NIR] : 6.345e-09
##
##           Kappa : 0.6565
##           Mcnemar's Test P-Value : 0.1213
##
##           Sensitivity : 0.9167
##           Specificity : 0.7317
##           Pos Pred Value : 0.8000
##           Neg Pred Value : 0.8824
##           Prevalence : 0.5393
##           Detection Rate : 0.4944
##           Detection Prevalence : 0.6180
##           Balanced Accuracy : 0.8242
##
##           'Positive' Class : X0
##
```

Lets include tunegrid with C parameters.

```
grid_radial <- expand.grid(sigma = c(0,0.01, 0.02, 0.025, 0.03, 0.04,
  0.05, 0.06, 0.07,0.08, 0.09, 0.1, 0.25, 0.5, 0.75,0.9),
  C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75,
  1, 1.5, 2,5))
set.seed(3233)
svm_Radial_Grid <- train(V14 ~., data = training, method = "svmRadial",
  trControl=trnctrl,
  preProcess = c("center", "scale"),
  tuneGrid = grid_radial,
  tuneLength = 10)
```

```
## Warning: model fit failed for Fold01.Rep1: sigma=0.000, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold01.Rep1: sigma=0.010, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold01.Rep1: sigma=0.020, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold01.Rep1: sigma=0.025, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold01.Rep1: sigma=0.030, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold01.Rep1: sigma=0.040, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold01.Rep1: sigma=0.050, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold01.Rep1: sigma=0.060, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]


```

## Warning: model fit failed for Fold10.Rep3: sigma=0.050, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold10.Rep3: sigma=0.060, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold10.Rep3: sigma=0.070, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold10.Rep3: sigma=0.080, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold10.Rep3: sigma=0.090, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold10.Rep3: sigma=0.100, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold10.Rep3: sigma=0.250, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold10.Rep3: sigma=0.500, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold10.Rep3: sigma=0.750, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning: model fit failed for Fold10.Rep3: sigma=0.900, C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results

```

svm_Radial_Grid

```

## Support Vector Machines with Radial Basis Function Kernel
##
## 211 samples
## 13 predictor
## 2 classes: 'X0', 'X1'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 190, 189, 190, 190, 190, 191, ...
## Resampling results across tuning parameters:
##
##   sigma  C      Accuracy  Kappa
##   0.000  0.00      NaN      NaN
##   0.000  0.01  0.5402958  0.000000000
##   0.000  0.05  0.5402958  0.000000000
##   0.000  0.10  0.5402958  0.000000000
##   0.000  0.25  0.5402958  0.000000000
##   0.000  0.50  0.5402958  0.000000000
##   0.000  0.75  0.5402958  0.000000000
##   0.000  1.00  0.5402958  0.000000000
##   0.000  1.50  0.5402958  0.000000000
##   0.000  2.00  0.5402958  0.000000000

```


##	0.000	5.00	0.5402958	0.000000000
##	0.010	0.00	NaN	NaN
##	0.010	0.01	0.5402958	0.000000000
##	0.010	0.05	0.5402958	0.000000000
##	0.010	0.10	0.7888240	0.561334844
##	0.010	0.25	0.8264502	0.646593548
##	0.010	0.50	0.8358369	0.665837183
##	0.010	0.75	0.8423304	0.680148558
##	0.010	1.00	0.8453608	0.686618362
##	0.010	1.50	0.8470202	0.690222637
##	0.010	2.00	0.8486147	0.693273878
##	0.010	5.00	0.8439177	0.684376167
##	0.020	0.00	NaN	NaN
##	0.020	0.01	0.5402958	0.000000000
##	0.020	0.05	0.6622583	0.277906265
##	0.020	0.10	0.8326551	0.658525222
##	0.020	0.25	0.8360606	0.666557360
##	0.020	0.50	0.8468759	0.689598160
##	0.020	0.75	0.8483838	0.692874219
##	0.020	1.00	0.8405916	0.677167082
##	0.020	1.50	0.8405916	0.677167082
##	0.020	2.00	0.8359019	0.667966203
##	0.020	5.00	0.8279582	0.652427542
##	0.025	0.00	NaN	NaN
##	0.025	0.01	0.5402958	0.000000000
##	0.025	0.05	0.7112049	0.389633308
##	0.025	0.10	0.8218326	0.637356769
##	0.025	0.25	0.8390837	0.672992244
##	0.025	0.50	0.8468759	0.689598160
##	0.025	0.75	0.8421068	0.680145607
##	0.025	1.00	0.8390043	0.674028850
##	0.025	1.50	0.8343867	0.665084932
##	0.025	2.00	0.8296176	0.655787095
##	0.025	5.00	0.8183550	0.632773713
##	0.030	0.00	NaN	NaN
##	0.030	0.01	0.5402958	0.000000000
##	0.030	0.05	0.7538023	0.485171353
##	0.030	0.10	0.8249351	0.643984403
##	0.030	0.25	0.8390837	0.673419794
##	0.030	0.50	0.8483838	0.692874219
##	0.030	0.75	0.8374170	0.670554672
##	0.030	1.00	0.8390043	0.674028850
##	0.030	1.50	0.8312843	0.658985748
##	0.030	2.00	0.8296176	0.655787095
##	0.030	5.00	0.8104113	0.616928143
##	0.040	0.00	NaN	NaN
##	0.040	0.01	0.5402958	0.000000000
##	0.040	0.05	0.7826263	0.547697491
##	0.040	0.10	0.8313636	0.657773685
##	0.040	0.25	0.8405195	0.676822102
##	0.040	0.50	0.8406638	0.677754139
##	0.040	0.75	0.8375613	0.671636109
##	0.040	1.00	0.8296970	0.655847517
##	0.040	1.50	0.8296176	0.655419169

##	0.040	2.00	0.8231169	0.642356941
##	0.040	5.00	0.8011688	0.597898954
##	0.050	0.00	NaN	NaN
##	0.050	0.01	0.5402958	0.000000000
##	0.050	0.05	0.7840620	0.550556916
##	0.050	0.10	0.8346176	0.664528756
##	0.050	0.25	0.8405916	0.677368985
##	0.050	0.50	0.8390765	0.674615907
##	0.050	0.75	0.8343867	0.665330460
##	0.050	1.00	0.8280303	0.652280937
##	0.050	1.50	0.8216017	0.639377143
##	0.050	2.00	0.8183478	0.632906929
##	0.050	5.00	0.7946681	0.585325348
##	0.060	0.00	NaN	NaN
##	0.060	0.01	0.5402958	0.000000000
##	0.060	0.05	0.7426768	0.460398108
##	0.060	0.10	0.8314430	0.658477979
##	0.060	0.25	0.8406638	0.677691287
##	0.060	0.50	0.8374892	0.671477281
##	0.060	0.75	0.8327201	0.662155604
##	0.060	1.00	0.8280303	0.652547431
##	0.060	1.50	0.8184993	0.633062445
##	0.060	2.00	0.8073016	0.610887497
##	0.060	5.00	0.7853535	0.566993276
##	0.070	0.00	NaN	NaN
##	0.070	0.01	0.5402958	0.000000000
##	0.070	0.05	0.6966811	0.358598823
##	0.070	0.10	0.8314430	0.658477979
##	0.070	0.25	0.8375685	0.671718420
##	0.070	0.50	0.8359019	0.668310258
##	0.070	0.75	0.8327201	0.662155604
##	0.070	1.00	0.8264430	0.649409604
##	0.070	1.50	0.8200794	0.637041559
##	0.070	2.00	0.8025253	0.601213539
##	0.070	5.00	0.7728716	0.541978817
##	0.080	0.00	NaN	NaN
##	0.080	0.01	0.5402958	0.000000000
##	0.080	0.05	0.6338095	0.217415445
##	0.080	0.10	0.8314430	0.658477979
##	0.080	0.25	0.8327994	0.662390751
##	0.080	0.50	0.8374098	0.671472319
##	0.080	0.75	0.8311328	0.659016715
##	0.080	1.00	0.8280303	0.652640867
##	0.080	1.50	0.8119048	0.620761402
##	0.080	2.00	0.7946609	0.585894158
##	0.080	5.00	0.7726479	0.542065602
##	0.090	0.00	NaN	NaN
##	0.090	0.01	0.5402958	0.000000000
##	0.090	0.05	0.6052886	0.149626730
##	0.090	0.10	0.8329582	0.661408928
##	0.090	0.25	0.8279582	0.652833294
##	0.090	0.50	0.8343074	0.665895454
##	0.090	0.75	0.8294661	0.655964965
##	0.090	1.00	0.8232612	0.643356130

##	0.090	1.50	0.8072150	0.611559221
##	0.090	2.00	0.7948052	0.586758024
##	0.090	5.00	0.7677994	0.533144683
##	0.100	0.00	NaN	NaN
##	0.100	0.01	0.5402958	0.000000000
##	0.100	0.05	0.5816811	0.096400169
##	0.100	0.10	0.8330303	0.661147548
##	0.100	0.25	0.8234127	0.644679158
##	0.100	0.50	0.8295455	0.656746603
##	0.100	0.75	0.8278788	0.652952445
##	0.100	1.00	0.8167532	0.630296880
##	0.100	1.50	0.8057720	0.608964351
##	0.100	2.00	0.7917027	0.581470597
##	0.100	5.00	0.7597835	0.517500042
##	0.250	0.00	NaN	NaN
##	0.250	0.01	0.5402958	0.000000000
##	0.250	0.05	0.5402958	0.000000000
##	0.250	0.10	0.5402958	0.000000000
##	0.250	0.25	0.7209091	0.419490547
##	0.250	0.50	0.7851154	0.575467881
##	0.250	0.75	0.7804329	0.562970809
##	0.250	1.00	0.7787662	0.557375920
##	0.250	1.50	0.7787662	0.556921206
##	0.250	2.00	0.7721861	0.543618523
##	0.250	5.00	0.7690837	0.538017156
##	0.500	0.00	NaN	NaN
##	0.500	0.01	0.5402958	0.000000000
##	0.500	0.05	0.5402958	0.000000000
##	0.500	0.10	0.5402958	0.000000000
##	0.500	0.25	0.5402958	0.000000000
##	0.500	0.50	0.5452092	0.012369398
##	0.500	0.75	0.6743434	0.325614816
##	0.500	1.00	0.7484271	0.501544595
##	0.500	1.50	0.7498629	0.502789023
##	0.500	2.00	0.7515296	0.506014524
##	0.500	5.00	0.7515296	0.506014524
##	0.750	0.00	NaN	NaN
##	0.750	0.01	0.5402958	0.000000000
##	0.750	0.05	0.5402958	0.000000000
##	0.750	0.10	0.5402958	0.000000000
##	0.750	0.25	0.5402958	0.000000000
##	0.750	0.50	0.5402958	0.000000000
##	0.750	0.75	0.5405195	0.001814767
##	0.750	1.00	0.6037590	0.158654468
##	0.750	1.50	0.6275180	0.215818373
##	0.750	2.00	0.6275180	0.215818373
##	0.750	5.00	0.6275180	0.215818373
##	0.900	0.00	NaN	NaN
##	0.900	0.01	0.5402958	0.000000000
##	0.900	0.05	0.5402958	0.000000000
##	0.900	0.10	0.5402958	0.000000000
##	0.900	0.25	0.5402958	0.000000000
##	0.900	0.50	0.5402958	0.000000000
##	0.900	0.75	0.5402958	0.000000000

```
## 0.900 1.00 0.5515584 0.032670546
## 0.900 1.50 0.5737951 0.087485055
## 0.900 2.00 0.5737951 0.087485055
## 0.900 5.00 0.5737951 0.087485055
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01 and C = 2.
```

Confusion Matrix and Statistics

```
test_pred_Radial_Grid <- predict(svm_Radial_Grid, newdata = testing)
confusionMatrix(test_pred_Radial_Grid, testing$V14 )
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction X0 X1
##           X0 44 11
##           X1  4 30
##
##              Accuracy : 0.8315
##              95% CI : (0.7373, 0.9025)
##      No Information Rate : 0.5393
##      P-Value [Acc > NIR] : 6.345e-09
##
##              Kappa : 0.6565
##  Mcnemar's Test P-Value : 0.1213
##
##      Sensitivity : 0.9167
##      Specificity : 0.7317
##      Pos Pred Value : 0.8000
##      Neg Pred Value : 0.8824
##      Prevalence : 0.5393
##      Detection Rate : 0.4944
##      Detection Prevalence : 0.6180
##      Balanced Accuracy : 0.8242
##
##      'Positive' Class : X0
##
```