Identifying Heart Disease using Support Vector Machine Algorithm

Objective

The objective is to build a SVM classifier using caret package to predict whether a patient is suffering from any heart disease or not.

Approach and Activities

The activities include data import, preparation, exploration, training data set with SVM algorithm using linear and rbf kernels, building the SVM model and test testing data, evaluating model performance and tuning model by identifying best hyperpameters for higher accuracy and generalization

Dataset Information

The dataset is about Heart Disease data set consists of 14 attributes data. All the attributes consist of numeric values. First 13 variables will be used for predicting 14th variables.The target variable is at index 14

Below is the data source link

http://dataaspirant.com/wp-content/uploads/2017/01/heart_tidy.csv

There is no headers provided in the data sheet as these are confidential information. However, following is the desciption of each column/variable as addition information

Feature Title: Variable Data: Type Feature Categorization

1. age: Continuous Variable: 29 - 77
2. sex: Categorical Variable: 1 = male; 0 = female
3. cp: chest pain type Categorical Variable: 1- typical angina 2-atypical angina 3-non-anginal pain 4- asymptomatic
4. trestbps: resting blood pressure Continuous Variable: 94 - 200

5.chol: serum cholestoral Continuous Variable: 126 - 564

6.fbs- fasting blood sugar> 120 mg/dl: Categorical Variable: 1 = true; 0 = false

7.restecg:resting ECG results Categorical Variable: 0- normal,1- having ST-T wave abnormality

8.thalach- maximum heart rate achieved: Continuous Variable: 71 - 202

9.exang- exercise-induced angina: Categorical Variable: 1 = yes; 0 = no

10. oldpeak-ST depression induced by exercise relative to rest: Continuous Variable: 0 - 6.2

11. slope: slope of the peak exercise ST segment: Continuous Variable:1 - 3

12. ca-number of major vessels Continuous: Variable 0 - 3

13. thal: Categorical Variable: 3 = normal;6 = fixed defect;7 = reversible defect

14 Target Variable: Categorical Variable: 0- Absence of Heart Disease, 1- Presence of Heart Disease

Library Importing and data reading

```
#Importing caret library
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.1
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.5.1
```

```
#Data reading
heart_df <- read.csv("heart_tidy.csv", sep = ',', header = FALSE)
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 ...
```

```
head(heart_df)
```

```
##   V1 V2 V3  V4  V5 V6 V7  V8 V9 V10 V11 V12 V13 V14
## 1 63  1  1 145 233  1  2 150  0 2.3   3   0   6   0
## 2 67  1  4 160 286  0  2 108  1 1.5   2   3   3   1
## 3 67  1  4 120 229  0  2 129  1 2.6   2   2   7   1
## 4 37  1  3 130 250  0  0 187  0 3.5   3   0   3   0
## 5 41  0  2 130 204  0  2 172  0 1.4   1   0   3   0
## 6 56  1  2 120 236  0  0 178  0 0.8   1   0   3   0
```

Dataset summarized details

```
anyNA(heart_df)
```

```
## [1] FALSE
```

```
summary(heart_df)
```

```
##        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
```

Data preparation - splitting data into training and test dataset

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

```
## [1] 210  14
```

```
dim(testing)
```

```
## [1] 90 14
```

```
#Converting training and testing data frame's "V14" column to factor variable
training[["V14"]] = factor(training[["V14"]])
```

Training the SVM model using linear kernal

```
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(3233)
svm_Linear <- train(V14 ~., data = training, method = "svmLinear",
                    trControl=trctrl,
                    preProcess = c("center", "scale"),
                    tuneLength = 10)

# Trained SVM model result
svm_Linear
```

```
## Support Vector Machines with Linear Kernel
##
## 210 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: 189, 189, 189, 189, 189, 189, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7920635  0.581696
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
# Test Set Prediction
test_pred <- predict(svm_Linear, testing)
test_pred
```

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

```
## [71] 0 1 1 0 1 1 0 0 0 1 1 1 1 0 1 0 0 0 1 0
## Levels: 0 1
```

```
# Confusion matrix and model perfromance metrics
# Converting testing data frame's "V14" column to factor variable
testing$V14=as.factor(testing$V14)
confusionMatrix(test_pred, testing$V14)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 45  5
##          1  7 33
##
##                Accuracy : 0.8667
##                  95% CI : (0.7787, 0.9292)
##     No Information Rate : 0.5778
##     P-Value [Acc > NIR] : 2.884e-09
##
##                   Kappa : 0.7286
##  Mcnemar's Test P-Value : 0.7728
##
##             Sensitivity : 0.8654
##             Specificity : 0.8684
##          Pos Pred Value : 0.9000
##          Neg Pred Value : 0.8250
##              Prevalence : 0.5778
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.5556
##       Balanced Accuracy : 0.8669
##
##        'Positive' Class : 0
##
```

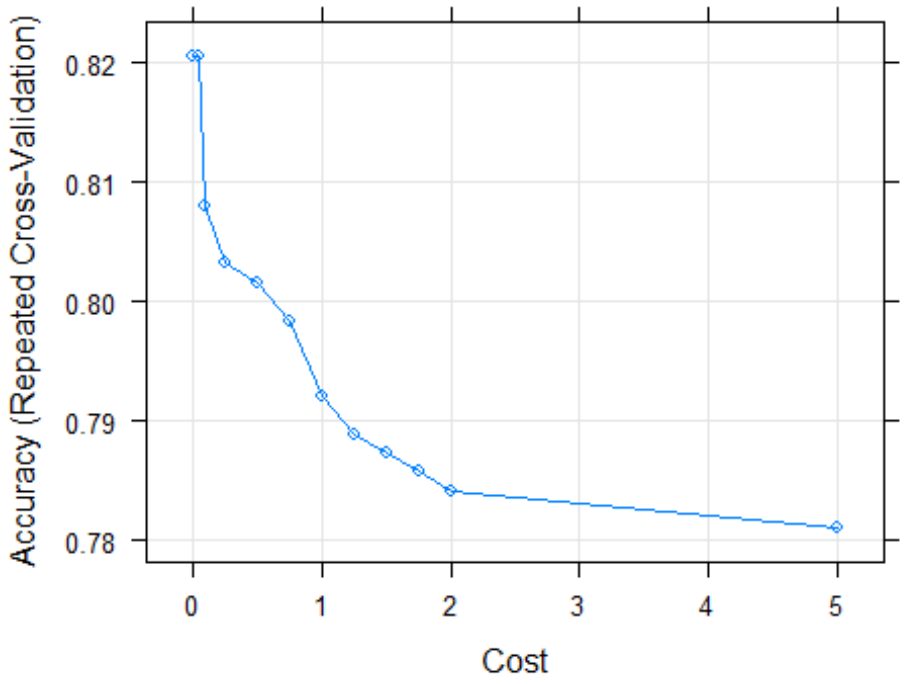
```
# Building & tuning of an SVM classifier with different values of C
# Customizations for selecting C value in Linear classifier
grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1
.75, 2,5))
set.seed(3233)
svm_Linear_Grid <- train(V14 ~., data = training, method = "svmLinear",
                    trControl=trctrl,
                    preProcess = c("center", "scale"),
                    tuneGrid = grid,
                    tuneLength = 10)

svm_Linear_Grid
```

```
## Support Vector Machines with Linear Kernel
##
## 210 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: 189, 189, 189, 189, 189, 189, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.00       NaN        NaN
##   0.01  0.8206349  0.6378332
##   0.05  0.8206349  0.6377730
##   0.10  0.8079365  0.6127299
##   0.25  0.8031746  0.6038409
##   0.50  0.8015873  0.6007601
##   0.75  0.7984127  0.5942518
##   1.00  0.7920635  0.5816960
##   1.25  0.7888889  0.5753318
##   1.50  0.7873016  0.5722214
##   1.75  0.7857143  0.5690837
##   2.00  0.7841270  0.5659745
##   5.00  0.7809524  0.5596986
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.01.
```

```
plot(svm_Linear_Grid)
```



```
#The above plot is showing that our classifier is giving best accuracy on C =
0.01. Following is  the predictions made using this model for test set.

test_pred_grid <- predict(svm_Linear_Grid, newdata = testing)
test_pred_grid
```

```
##  [1] 0 1 1 1 0 0 1 0 0 1 0 1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0
## [36] 1 1 1 1 1 1 0 0 1 0 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0
## [71] 0 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 0 0 1 0
## Levels: 0 1
```

```
#Model perfromance metrics
confusionMatrix(test_pred_grid, testing$V14 )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 46  6
##          1  6 32
##
##                Accuracy : 0.8667
##                  95% CI : (0.7787, 0.9292)
##     No Information Rate : 0.5778
##     P-Value [Acc > NIR] : 2.884e-09
##
##                   Kappa : 0.7267
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8846
##             Specificity : 0.8421
##          Pos Pred Value : 0.8846
##          Neg Pred Value : 0.8421
##              Prevalence : 0.5778
##          Detection Rate : 0.5111
##    Detection Prevalence : 0.5778
##       Balanced Accuracy : 0.8634
##
##        'Positive' Class : 0
##
```

```
# The results of confusion matrix show that the accuracy on the test set is 8
6.67 %.
```

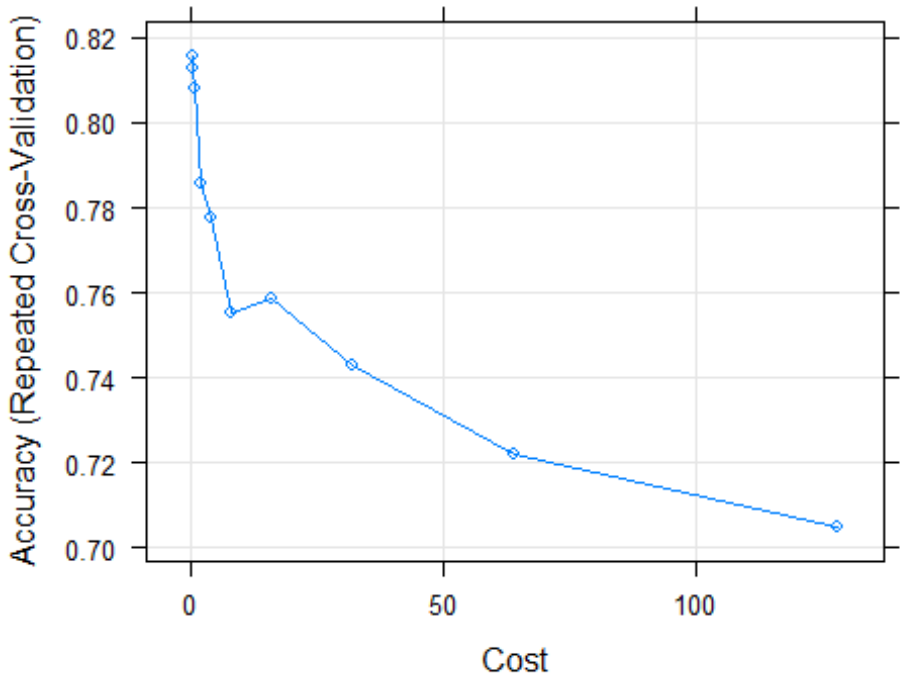
SVM Classifier using Non-Linear Kernel-Radial Basis Function (rbf)

```
#Building a model using Non-Linear Kernel- Radial Basis Function
set.seed(3233)
svm_Radial <- train(V14 ~., data = training, method = "svmRadial",
  trControl=trctrl,
  preProcess = c("center", "scale"),
```

```
  tuneLength = 10)
svm_Radial

## Support Vector Machines with Radial Basis Function Kernel
##
## 210 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: 189, 189, 189, 189, 189, 189, ...
## Resampling results across tuning parameters:
##
##   C       Accuracy   Kappa
##     0.25  0.8158730  0.6280984
##     0.50  0.8126984  0.6223160
##     1.00  0.8079365  0.6125220
##     2.00  0.7857143  0.5685607
##     4.00  0.7777778  0.5530706
##     8.00  0.7555556  0.5086964
##    16.00  0.7587302  0.5159259
##    32.00  0.7428571  0.4838627
##    64.00  0.7222222  0.4435755
##   128.00  0.7047619  0.4087328
##
## Tuning parameter 'sigma' was held constant at a value of 0.04250183
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.04250183 and C = 0.25.

plot(svm_Radial)
```




```
#It is showing that final sigma parameter's value is 0.04250183 & C parameter
's value as 0.25. Now, testing model's accuracy on the test set

#Test Set Prediction using learned rbf model
test_pred_Radial <- predict(svm_Radial, newdata = testing)
confusionMatrix(test_pred_Radial, testing$V14 )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 47  6
##          1  5 32
##
##                Accuracy : 0.8778
##                  95% CI : (0.7918, 0.9374)
##     No Information Rate : 0.5778
##     P-Value [Acc > NIR] : 5.854e-10
##
##                   Kappa : 0.7486
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9038
##             Specificity : 0.8421
##          Pos Pred Value : 0.8868
##          Neg Pred Value : 0.8649
##              Prevalence : 0.5778
```

```
##          Detection Rate : 0.5222
##    Detection Prevalence : 0.5889
##       Balanced Accuracy : 0.8730
##
##        'Positive' Class : 0
##
```

```
#Getting an accuracy of 87.78% with values of C=0.25 & sigma= 0.04250183

# Building & tuning of an SVM classifier with different values of C and sigma
# Customizations for selecting C and sigma value in rbf classifier

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=trctrl,
                   preProcess = c("center", "scale"),
                   tuneGrid = grid_radial,
                   tuneLength = 10)

svm_Radial_Grid
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 210 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: 189, 189, 189, 189, 189, 189, ...
## Resampling results across tuning parameters:
##
##   sigma  C     Accuracy   Kappa
##   0.000  0.00        NaN         NaN
##   0.000  0.01  0.5238095  0.000000000
##   0.000  0.05  0.5238095  0.000000000
##   0.000  0.10  0.5238095  0.000000000
##   0.000  0.25  0.5238095  0.000000000
##   0.000  0.50  0.5238095  0.000000000
##   0.000  0.75  0.5238095  0.000000000
##   0.000  1.00  0.5238095  0.000000000
##   0.000  1.50  0.5238095  0.000000000
##   0.000  2.00  0.5238095  0.000000000
##   0.000  5.00  0.5238095  0.000000000
##   0.010  0.00        NaN         NaN
##   0.010  0.01  0.5238095  0.000000000
##   0.010  0.05  0.5238095  0.000000000
```

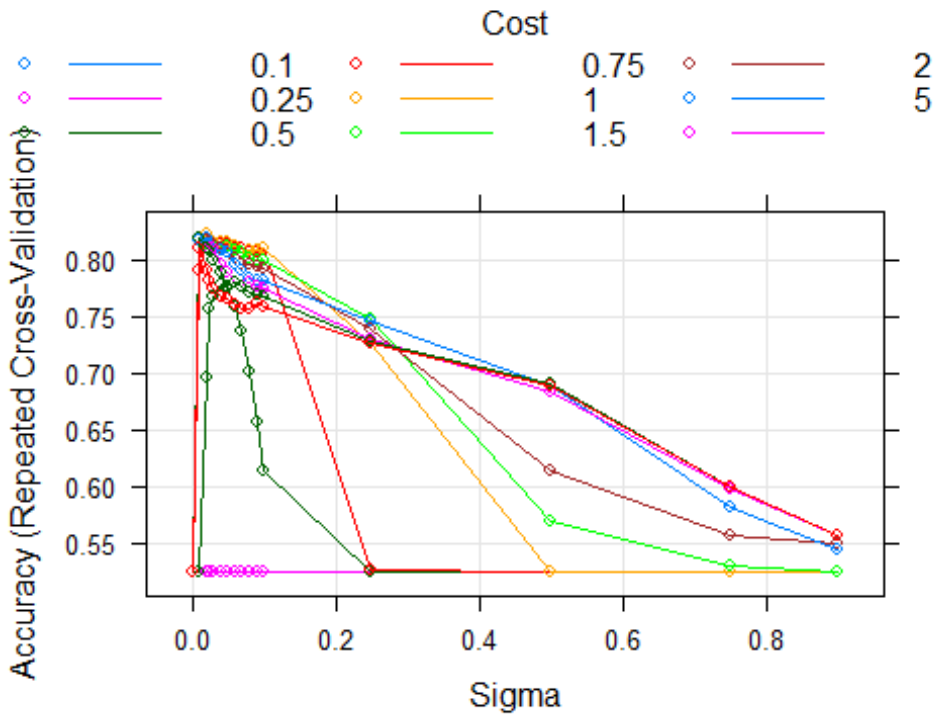
```
##   0.010  0.10  0.7904762  0.572773277
##   0.010  0.25  0.8111111  0.617503180
##   0.010  0.50  0.8190476  0.634519523
##   0.010  0.75  0.8174603  0.631528851
##   0.010  1.00  0.8174603  0.631352905
##   0.010  1.50  0.8190476  0.634666550
##   0.010  2.00  0.8190476  0.634666550
##   0.010  5.00  0.8111111  0.619121228
##   0.020  0.00        NaN          NaN
##   0.020  0.01  0.5238095  0.000000000
##   0.020  0.05  0.6952381  0.370309451
##   0.020  0.10  0.8174603  0.630021936
##   0.020  0.25  0.8222222  0.641206267
##   0.020  0.50  0.8142857  0.625078184
##   0.020  0.75  0.8174603  0.631528851
##   0.020  1.00  0.8190476  0.634722788
##   0.020  1.50  0.8111111  0.618915958
##   0.020  2.00  0.8095238  0.615630414
##   0.020  5.00  0.7904762  0.578177897
##   0.025  0.00        NaN          NaN
##   0.025  0.01  0.5238095  0.000000000
##   0.025  0.05  0.7571429  0.501717569
##   0.025  0.10  0.8158730  0.627180600
##   0.025  0.25  0.8158730  0.628156814
##   0.025  0.50  0.8142857  0.625105376
##   0.025  0.75  0.8158730  0.628419020
##   0.025  1.00  0.8142857  0.625368228
##   0.025  1.50  0.8095238  0.615630414
##   0.025  2.00  0.8079365  0.612375952
##   0.025  5.00  0.7825397  0.562431414
##   0.030  0.00        NaN          NaN
##   0.030  0.01  0.5238095  0.000000000
##   0.030  0.05  0.7666667  0.522491751
##   0.030  0.10  0.8158730  0.627327897
##   0.030  0.25  0.8158730  0.628215207
##   0.030  0.50  0.8126984  0.621968353
##   0.030  0.75  0.8142857  0.625280394
##   0.030  1.00  0.8111111  0.618944058
##   0.030  1.50  0.8095238  0.615630414
##   0.030  2.00  0.8000000  0.596829329
##   0.030  5.00  0.7746032  0.546420219
##   0.040  0.00        NaN          NaN
##   0.040  0.01  0.5238095  0.000000000
##   0.040  0.05  0.7777778  0.546099884
##   0.040  0.10  0.8142857  0.624457829
##   0.040  0.25  0.8158730  0.628098443
##   0.040  0.50  0.8111111  0.619061556
##   0.040  0.75  0.8095238  0.615862243
##   0.040  1.00  0.8079365  0.612522040
##   0.040  1.50  0.7968254  0.590465952
```

```
##   0.040  2.00  0.7888889  0.574894377
##   0.040  5.00  0.7682540  0.533689992
##   0.050  0.00        NaN          NaN
##   0.050  0.01  0.5238095  0.000000000
##   0.050  0.05  0.7746032  0.539466957
##   0.050  0.10  0.8142857  0.624780963
##   0.050  0.25  0.8158730  0.628098443
##   0.050  0.50  0.8126984  0.622345065
##   0.050  0.75  0.8063492  0.609411690
##   0.050  1.00  0.8063492  0.609383154
##   0.050  1.50  0.7888889  0.574836119
##   0.050  2.00  0.7761905  0.549556877
##   0.050  5.00  0.7650794  0.527295716
##   0.060  0.00        NaN          NaN
##   0.060  0.01  0.5238095  0.000000000
##   0.060  0.05  0.7587302  0.506304402
##   0.060  0.10  0.8111111  0.618300573
##   0.060  0.25  0.8126984  0.621881063
##   0.060  0.50  0.8079365  0.612610195
##   0.060  0.75  0.8015873  0.599880768
##   0.060  1.00  0.7952381  0.587007531
##   0.060  1.50  0.7793651  0.555802949
##   0.060  2.00  0.7793651  0.556185239
##   0.060  5.00  0.7603175  0.518483308
##   0.070  0.00        NaN          NaN
##   0.070  0.01  0.5238095  0.000000000
##   0.070  0.05  0.7365079  0.458840485
##   0.070  0.10  0.8111111  0.618330431
##   0.070  0.25  0.8063492  0.609094732
##   0.070  0.50  0.8063492  0.609443172
##   0.070  0.75  0.7952381  0.587095518
##   0.070  1.00  0.7904762  0.577855206
##   0.070  1.50  0.7761905  0.549499422
##   0.070  2.00  0.7761905  0.549790799
##   0.070  5.00  0.7571429  0.512342204
##   0.080  0.00        NaN          NaN
##   0.080  0.01  0.5238095  0.000000000
##   0.080  0.05  0.7015873  0.384579214
##   0.080  0.10  0.8063492  0.608623691
##   0.080  0.25  0.8079365  0.612787335
##   0.080  0.50  0.8015873  0.599970753
##   0.080  0.75  0.7952381  0.587185098
##   0.080  1.00  0.7841270  0.565155380
##   0.080  1.50  0.7793651  0.556038205
##   0.080  2.00  0.7714286  0.540317989
##   0.080  5.00  0.7571429  0.512520301
##   0.090  0.00        NaN          NaN
##   0.090  0.01  0.5238095  0.000000000
##   0.090  0.05  0.6571429  0.289294589
##   0.090  0.10  0.8063492  0.608476664
```

```
##   0.090  0.25  0.8079365  0.613251126
##   0.090  0.50  0.8000000  0.596745743
##   0.090  0.75  0.7936508  0.584014883
##   0.090  1.00  0.7809524  0.558792544
##   0.090  1.50  0.7761905  0.549791599
##   0.090  2.00  0.7698413  0.537236447
##   0.090  5.00  0.7619048  0.522375325
##   0.100  0.00        NaN          NaN
##   0.100  0.01  0.5238095  0.000000000
##   0.100  0.05  0.6142857  0.197118299
##   0.100  0.10  0.8095238  0.614631322
##   0.100  0.25  0.8111111  0.619993590
##   0.100  0.50  0.8000000  0.597123554
##   0.100  0.75  0.7920635  0.580877185
##   0.100  1.00  0.7809524  0.559381191
##   0.100  1.50  0.7746032  0.546822855
##   0.100  2.00  0.7682540  0.534499910
##   0.100  5.00  0.7587302  0.516073614
##   0.250  0.00        NaN          NaN
##   0.250  0.01  0.5238095  0.000000000
##   0.250  0.05  0.5238095  0.000000000
##   0.250  0.10  0.5253968  0.003475513
##   0.250  0.25  0.7253968  0.439136882
##   0.250  0.50  0.7476190  0.496851555
##   0.250  0.75  0.7380952  0.477159921
##   0.250  1.00  0.7460317  0.491597676
##   0.250  1.50  0.7301587  0.460075036
##   0.250  2.00  0.7285714  0.456789050
##   0.250  5.00  0.7253968  0.450540095
##   0.500  0.00        NaN          NaN
##   0.500  0.01  0.5238095  0.000000000
##   0.500  0.05  0.5238095  0.000000000
##   0.500  0.10  0.5238095  0.000000000
##   0.500  0.25  0.5238095  0.000000000
##   0.500  0.50  0.5682540  0.098762428
##   0.500  0.75  0.6126984  0.204503243
##   0.500  1.00  0.6888889  0.379763899
##   0.500  1.50  0.6841270  0.371094398
##   0.500  2.00  0.6904762  0.384000578
##   0.500  5.00  0.6888889  0.380945645
##   0.750  0.00        NaN          NaN
##   0.750  0.01  0.5238095  0.000000000
##   0.750  0.05  0.5238095  0.000000000
##   0.750  0.10  0.5238095  0.000000000
##   0.750  0.25  0.5238095  0.000000000
##   0.750  0.50  0.5301587  0.013902054
##   0.750  0.75  0.5571429  0.074827171
##   0.750  1.00  0.5809524  0.138239103
##   0.750  1.50  0.5968254  0.174064271
##   0.750  2.00  0.5984127  0.177738543
```

```
##   0.750  5.00  0.5984127  0.177738543
##   0.900  0.00         NaN          NaN
##   0.900  0.01  0.5238095  0.000000000
##   0.900  0.05  0.5238095  0.000000000
##   0.900  0.10  0.5238095  0.000000000
##   0.900  0.25  0.5238095  0.000000000
##   0.900  0.50  0.5238095  0.000000000
##   0.900  0.75  0.5492063  0.055825807
##   0.900  1.00  0.5444444  0.055132187
##   0.900  1.50  0.5555556  0.081488190
##   0.900  2.00  0.5555556  0.081488190
##   0.900  5.00  0.5555556  0.081488190
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.02 and C = 0.25.
```

```
plot(svm_Radial_Grid)
```




```
#Test Set Prediction using  rbf algorithm with sigma = 0.02 and C = 0.25 and
evaluating performance metrics

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

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

```
## Prediction  0  1
##          0 47  6
##          1  5 32
##
##                Accuracy : 0.8778
##                  95% CI : (0.7918, 0.9374)
##     No Information Rate : 0.5778
##     P-Value [Acc > NIR] : 5.854e-10
##
##                   Kappa : 0.7486
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9038
##             Specificity : 0.8421
##          Pos Pred Value : 0.8868
##          Neg Pred Value : 0.8649
##              Prevalence : 0.5778
##          Detection Rate : 0.5222
##    Detection Prevalence : 0.5889
##       Balanced Accuracy : 0.8730
##
##        'Positive' Class : 0
##

#For  svm_Radial_Grid classifier, it's giving an accuracy of 87.78%. So, it s
hows Linear classifier  is not giving better results as compared to Radial cl
assifier  even after tuning it.
```