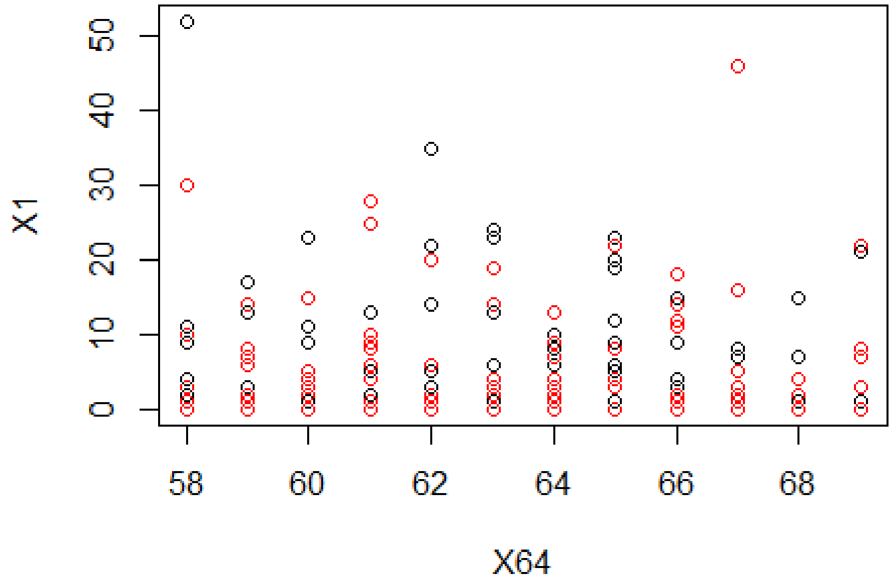
M6_L3_RomilShah

Romil Shah

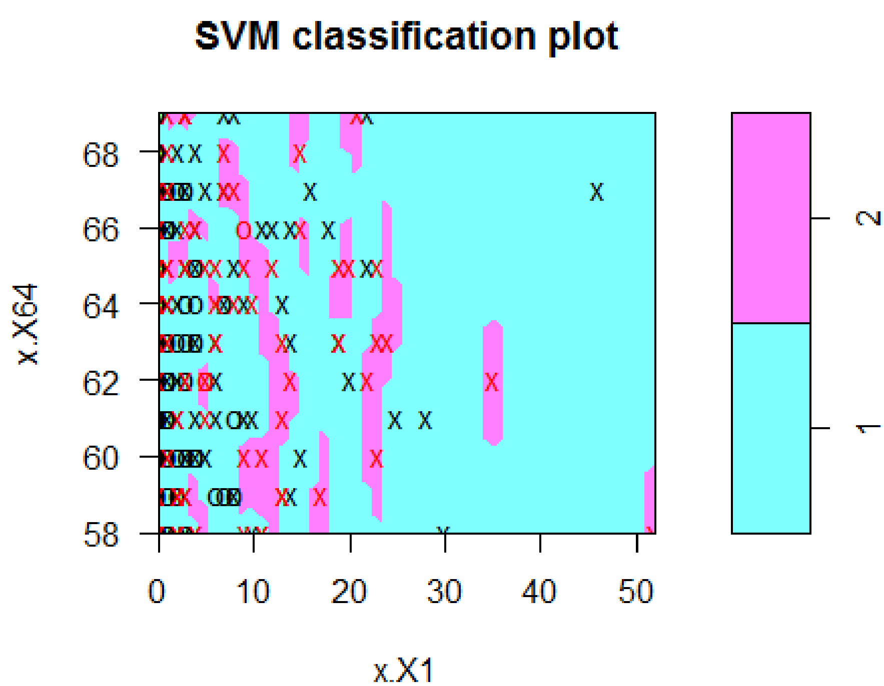
July 8, 2016

Read Data and additional packages

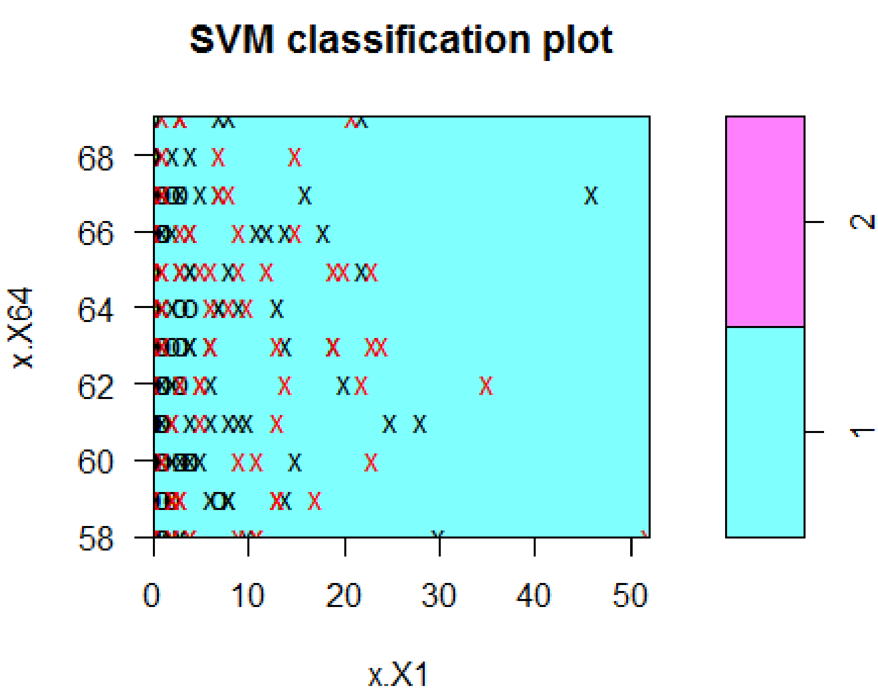
```
require(ggplot2)

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.2.5

require(e1071)

## Loading required package: e1071

## Warning: package 'e1071' was built under R version 3.2.5

require(kernlab)

## Loading required package: kernlab

## Warning: package 'kernlab' was built under R version 3.2.4

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha

# Haberman's survival
data_url <- 'https://archive.ics.uci.edu/ml/machine-learning-
databases/haberman/haberman.data'
dataframe <- read.csv(url(data_url), sep=",", header = TRUE)
temp=dataframe
temp$X1.1<-NULL
temp$X30<-NULL
x<-temp
temp=dataframe
y<-dataframe$X1.1
dataframe<-data.frame(x=x,y=as.factor(y))
plot(x,col=(3-y))
```




```r
# Training of model
svm.fit<- svm(y~.,data=dataframe, kernal="linear",cost=10,scale=FALSE)
plot(svm.fit,dataframe)
```



```r
summary(svm.fit)

##
## Call:
## svm(formula = y ~ ., data = dataframe, kernal = "linear", cost = 10,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.5
##
## Number of Support Vectors:  167
##
##  ( 88 79 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2

# Changing cost
svm.fit1<- svm(y~.,data=dataframe, kernal="linear",cost=0.1,scale=FALSE)
plot(svm.fit1,dataframe)
```



```r
tune.out<-
tune(svm,y~.,data=dataframe,kernal="linear",ranges=list(cost=c(0.001,0.01,0.1
,1,5,10,100)))
bestmodel<-tune.out$best.model
summary(bestmodel)
```

```
##
## Call:
## best.tune(method = svm, train.x = y ~ ., data = dataframe, ranges =
list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernal = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.001
##       gamma:  0.5
##
## Number of Support Vectors:  162
##
##  ( 81 81 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2
```

```r
# Predict test dataframe
testdf <- dataframe[1:25,]
testdf
```

```
##    x.X64 x.X1 y
## 1     62    3 1
## 2     65    0 1
## 3     59    2 1
## 4     65    4 1
## 5     58   10 1
## 6     60    0 1
## 7     59    0 2
## 8     66    9 2
## 9     58   30 1
## 10    60    1 1
## 11    61   10 1
## 12    67    7 1
## 13    60    0 1
## 14    64   13 1
## 15    63    0 1
## 16    60    1 1
## 17    69    0 1
```

```
## 18    60    0 1
## 19    63    0 1
## 20    58    0 1
## 21    59    6 1
## 22    60   15 1
## 23    63    0 1
## 24    69   21 2
## 25    59    2 1
```

```r
ypred=predict(bestmodel,testdf)
ypred
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
##  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## Levels: 1 2
```

```r
table(pred=ypred,truth=testdf$y)
```

```
##     truth
## pred  1  2
##    1 22  3
##    2  0  0
```

```r
agreement<-ypred==testdf$y
table(agreement)
```

```
## agreement
## FALSE  TRUE
##     3    22
```

```r
prop.table(table(agreement))
```

```
## agreement
## FALSE  TRUE
##  0.12  0.88
```

```r
# Improve model performance
classifier_rbf<-ksvm(y~.,data=testdf,kernel="rbfdot")
predictions_rbf<-predict(classifier_rbf,testdf)
predictions_rbf
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
## Levels: 1 2
```

```r
agreement_rbf<-predictions_rbf==testdf$y
table(agreement_rbf)
```

```
## agreement_rbf
## FALSE  TRUE
##     2    23
```

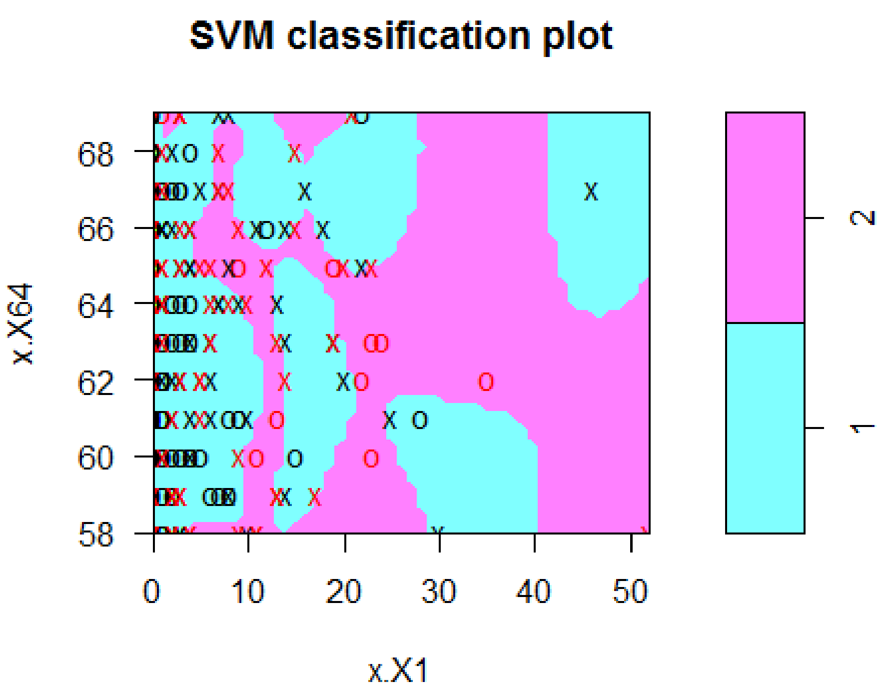
```r
prop.table(table(agreement_rbf))
```

```
## agreement_rbf
## FALSE  TRUE
##  0.08  0.92
```

```r
# Radial Kernal
svmfit1<-svm(y~.,data=dataframe,kernal="radial",gamma=1,cost=10000)
svmfit1
```

```
##
## Call:
## svm(formula = y ~ ., data = dataframe, kernal = "radial", gamma = 1,
##     cost = 10000)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10000
##       gamma:  1
##
## Number of Support Vectors:  171
```

```r
plot(svmfit1,dataframe)
```




```r
# Train model
tune.out=
tune(svm,y~.,data=dataframe,kernal="radial",ranges=list(cost=c(0.1,10,100,100
```

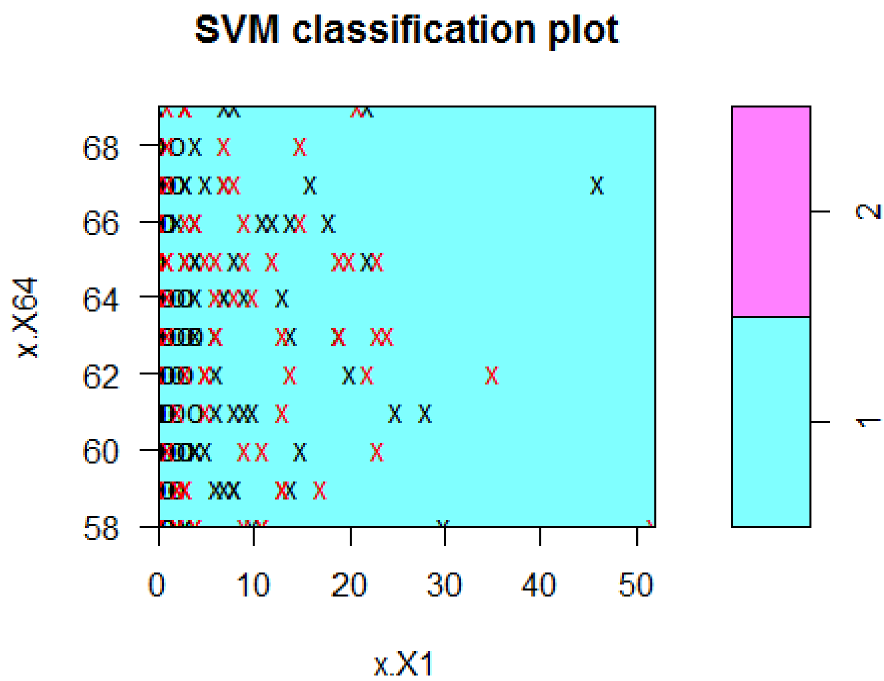
```
0)),gamma=c(0.5,1,2,3,4))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.2658065
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-01 0.2658065 0.05541501
## 2 1e+01 0.2821505 0.06622624
## 3 1e+02 0.3017204 0.05958063
## 4 1e+03 0.2854839 0.06619917
```

```
bestmodel1<-tune.out$best.model
bestmodel1
```

```
##
## Call:
## best.tune(method = svm, train.x = y ~ ., data = dataframe, ranges =
list(cost = c(0.1,
##     10, 100, 1000)), kernal = "radial", gamma = c(0.5, 1, 2,
##     3, 4))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.1
##       gamma:  0.5 1 2 3 4
##
## Number of Support Vectors:  170
```

```
plot(bestmodel1,dataframe)
```




```
# Predict test dataframe
testdf <- dataframe[1:25,]
testdf

##    x.X64 x.X1 y
## 1     62    3 1
## 2     65    0 1
## 3     59    2 1
## 4     65    4 1
## 5     58   10 1
## 6     60    0 1
## 7     59    0 2
## 8     66    9 2
## 9     58   30 1
## 10    60    1 1
## 11    61   10 1
## 12    67    7 1
## 13    60    0 1
## 14    64   13 1
## 15    63    0 1
## 16    60    1 1
## 17    69    0 1
## 18    60    0 1
## 19    63    0 1
## 20    58    0 1
## 21    59    6 1
## 22    60   15 1
```

```
## 23    63    0 1
## 24    69   21 2
## 25    59    2 1
```

```
ypred=predict(bestmodel1,testdf)
ypred
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
##  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## Levels: 1 2
```

```
table(pred=ypred,truth=testdf$y)
```

```
##     truth
## pred  1  2
##    1 22  3
##    2  0  0
```

```
agreement_radial<-ypred==testdf$y
table(agreement_radial)
```

```
## agreement_radial
## FALSE  TRUE
##     3    22
```

```
prop.table(table(agreement_radial))
```

```
## agreement_radial
## FALSE  TRUE
##  0.12  0.88
```

```
# Improve model performance
classifier_radial_rbf<-ksvm(y~.,data=testdf,kernel="rbfdot")
predictions_radial_rbf<-predict(classifier_radial_rbf,testdf)
predictions_radial_rbf
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
## Levels: 1 2
```

```
agreement_radial_rbf<-predictions_radial_rbf==testdf$y
table(agreement_radial_rbf)
```

```
## agreement_radial_rbf
## FALSE  TRUE
##     2    23
```

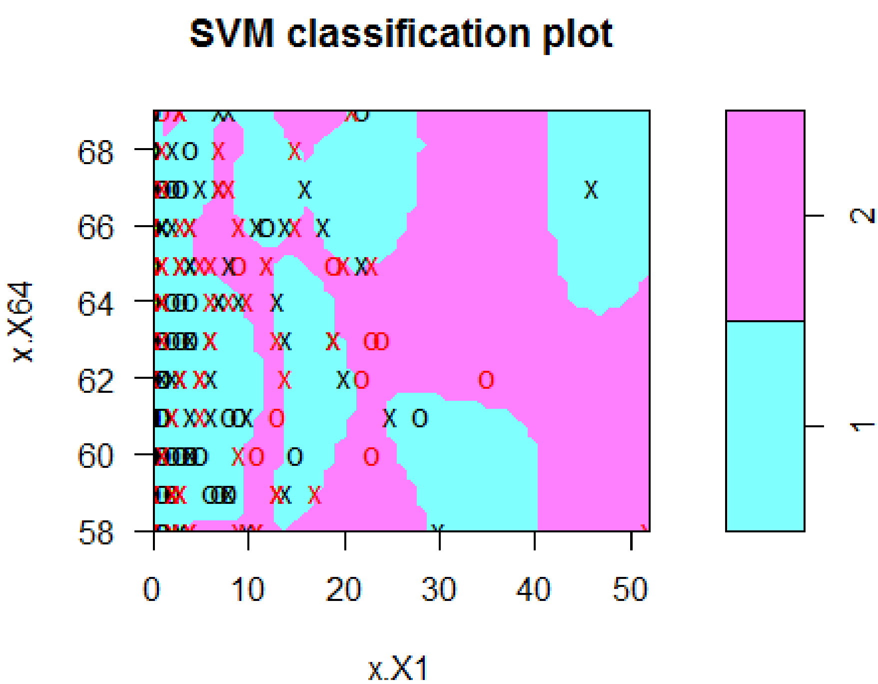
```
prop.table(table(agreement_radial_rbf))
```

```
## agreement_radial_rbf
## FALSE  TRUE
##  0.08  0.92
```

```r
# Polynomial Kernal
svmfit2<-svm(y~.,data=dataframe,kernal="polynomial",gamma=1,cost=10000)
svmfit2
```

```
##
## Call:
## svm(formula = y ~ ., data = dataframe, kernal = "polynomial",
##     gamma = 1, cost = 10000)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10000
##       gamma:  1
##
## Number of Support Vectors:  171
```

```r
plot(svmfit2,dataframe)
```




```r
# Train model
tune.out=
tune(svm,y~.,data=dataframe,kernal="polynomial",ranges=list(cost=c(0.1,10,100
,1000)),gamma=c(0.5,1,2,3,4))
summary(tune.out)
```

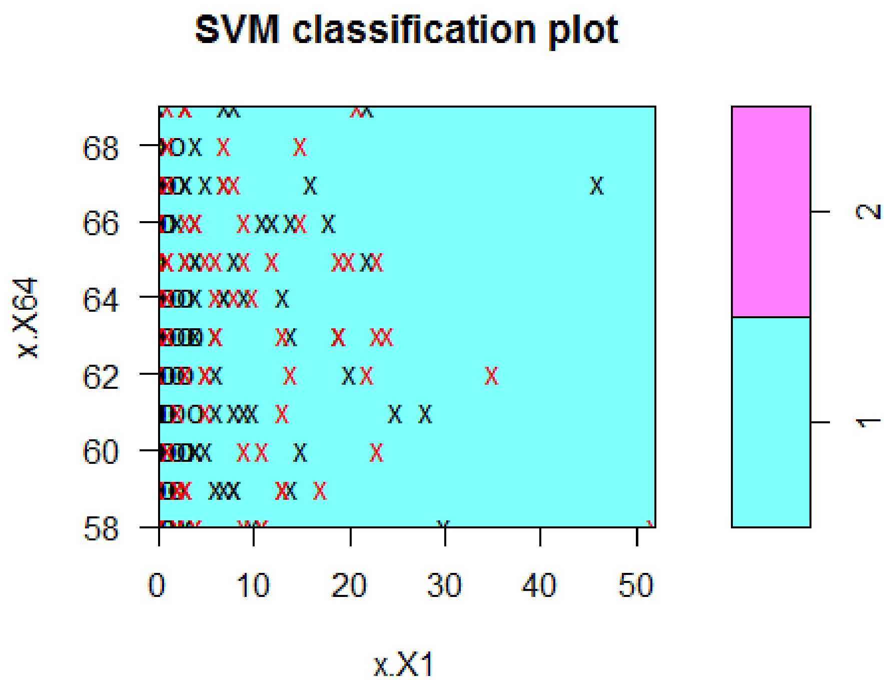
```
##
## Parameter tuning of 'svm':
```

```
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.2658065
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-01 0.2658065 0.07025967
## 2 1e+01 0.2852688 0.07623724
## 3 1e+02 0.2951613 0.06029660
## 4 1e+03 0.3051613 0.07976147
```

```
bestmodel2<-tune.out$best.model
bestmodel2
```

```
##
## Call:
## best.tune(method = svm, train.x = y ~ ., data = dataframe, ranges =
list(cost = c(0.1,
##     10, 100, 1000)), kernal = "polynomial", gamma = c(0.5, 1,
##     2, 3, 4))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.1
##       gamma:  0.5 1 2 3 4
##
## Number of Support Vectors:  170
```

```
plot(bestmodel2,dataframe)
```




```
# Predict test dataframe
testdf <- dataframe[1:25,]
testdf
```

```
##    x.X64 x.X1 y
## 1     62    3 1
## 2     65    0 1
## 3     59    2 1
## 4     65    4 1
## 5     58   10 1
## 6     60    0 1
## 7     59    0 2
## 8     66    9 2
## 9     58   30 1
## 10    60    1 1
## 11    61   10 1
## 12    67    7 1
## 13    60    0 1
## 14    64   13 1
## 15    63    0 1
## 16    60    1 1
## 17    69    0 1
## 18    60    0 1
## 19    63    0 1
## 20    58    0 1
## 21    59    6 1
## 22    60   15 1
```

```
## 23     63     0 1
## 24     69    21 2
## 25     59     2 1
```

```
ypred=predict(bestmodel2,testdf)
ypred
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
##  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## Levels: 1 2
```

```
table(pred=ypred,truth=testdf$y)
```

```
##     truth
## pred  1  2
##    1 22  3
##    2  0  0
```

```
agreement_poly<-ypred==testdf$y
table(agreement_poly)
```

```
## agreement_poly
## FALSE  TRUE
##     3    22
```

```
prop.table(table(agreement_poly))
```

```
## agreement_poly
## FALSE  TRUE
##  0.12  0.88
```

```
# Improve model performance
classifier_poly_rbf<-ksvm(y~.,data=testdf,kernel="rbfdot")
predictions_poly_rbf<-predict(classifier_poly_rbf,testdf)
predictions_poly_rbf
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
## Levels: 1 2
```

```
agreement_poly_rbf<-predictions_poly_rbf==testdf$y
table(agreement_poly_rbf)
```

```
## agreement_poly_rbf
## FALSE  TRUE
##     2    23
```

```
prop.table(table(agreement_poly_rbf))
```

```
## agreement_poly_rbf
## FALSE  TRUE
##  0.08  0.92
```

Answers:

A(1)

The linear classifier has the efficiency of true and false is as follows:

```
prop.table(table(agreement))

## agreement
## FALSE  TRUE
##  0.12  0.88
```

The radial classifier has the efficiency of true and false is as follows:

```
prop.table(table(agreement_radial))

## agreement_radial
## FALSE  TRUE
##  0.12  0.88
```

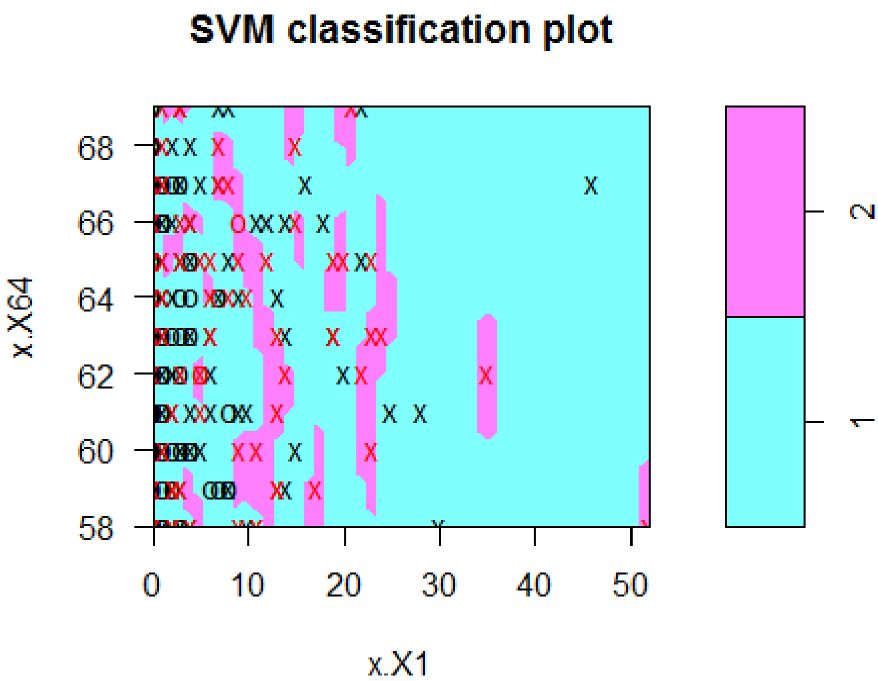
It is seen that they are quite similar in the output. Thus for this database, the classifiers perform pretty well with over 0.88 TRUE and 0.12 FALSE for both linear and radial kernal. By changing cost and gamma, the best models give the efficiency of 0.92 TRUE and 0.08 FLASE asnd hence increases the TRUE values.

```
prop.table(table(agreement))

## agreement
## FALSE  TRUE
##  0.12  0.88

prop.table(table(agreement_radial))

## agreement_radial
## FALSE  TRUE
##  0.12  0.88
```

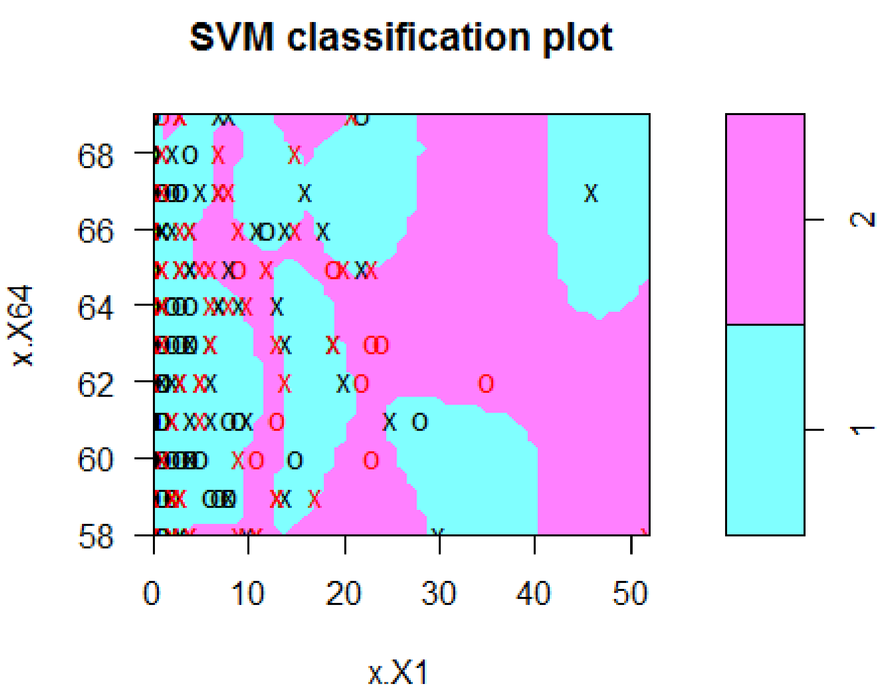
A(2)

Trying different kernals including linear, radial and polynomial and found that there is no change in the performance. All the kernals perform in the same way for this particular dataset. In terms of svm models, the linear model gives a clear and distinct graph in comaprison to that of polynomial and radial. Polynomial and radial have the same plot for the svm classifier and they are more spread out compared to linear.

```
#Linear Plot
plot(svm.fit,dataframe)
```




```r
#Radial Plot
plot(svmfit1,dataframe)
```



```r
#Polynomial Plot
plot(svmfit2,dataframe)
```

A(3)

Increasing the sample size and bringing in more data would improve the performance of the svm classifier. Also using ksvm with kernal 'rbfdot' will improve the performance. It is clearly seen that it does improve the performace from 0.88 TRUE to 0.92 TRUE thus by 4%.

```r
prop.table(table(agreement_rbf))
```

```
## agreement_rbf
## FALSE  TRUE
##  0.08  0.92
```

```r
prop.table(table(agreement_radial_rbf))
```

```
## agreement_radial_rbf
## FALSE  TRUE
##  0.08  0.92
```