HW9

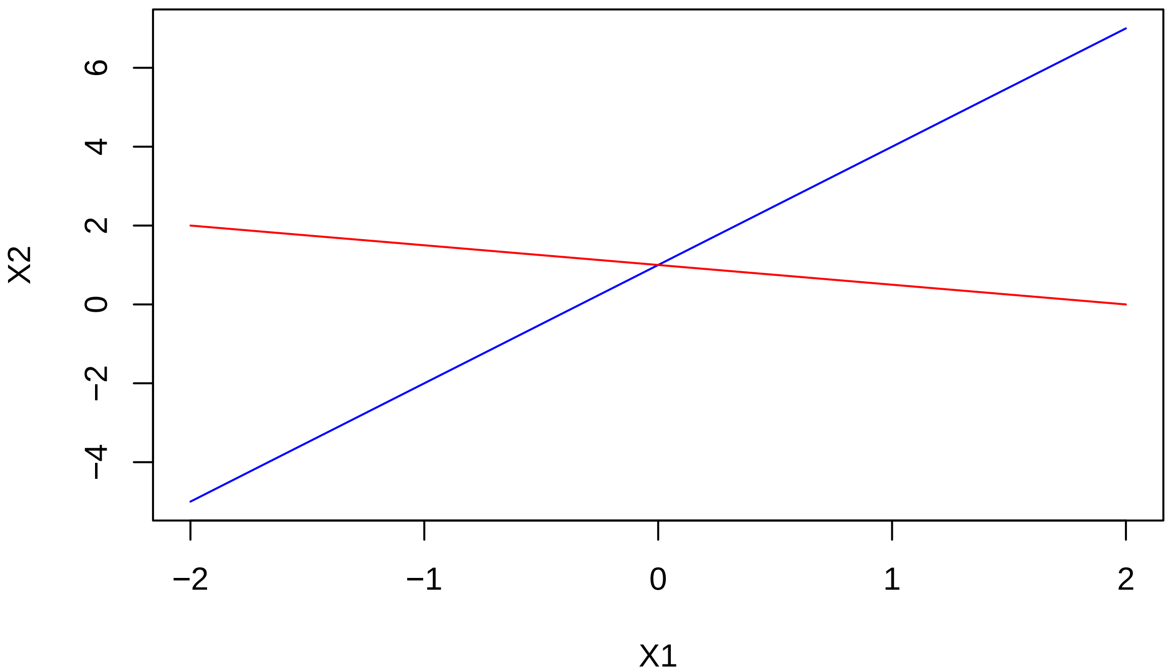
Kipling Stopa A15851786, Nathan Brodie A15874575, Shrest Venkatraman A15910171

2023-03-16

CONTRIBUTIONS All 3 team members worked on each question together, and helped to format the final RMarkdown Document. Work was done on Zoom collectively with equal contribution from each member.

##Question 1

```
SVM_test = read.csv("D:/downloads/SVM_test.csv")
SVM_train = read.csv("D:/downloads/SVM_train.csv")

library(e1071)
x = seq(-2,2,.2)
y = 3*x + 1
y_b = -1*x/2 + 1
plot(x,y,"l",col="blue",ylab="X2",xlab="X1")
lines(x,y_b,col="red")
```



##a) The Points above the blue line represent the points for which (1 + 3X1 - X2)<0, The points below the blue line represent the points for which (1+3X1 - X2)>0. ##b) The Points above the red line represent the points for which (-2 + X1 + 2*X2)<0, The points below the red line represent the points for which (-2 + X1 + 2*X2)>0.

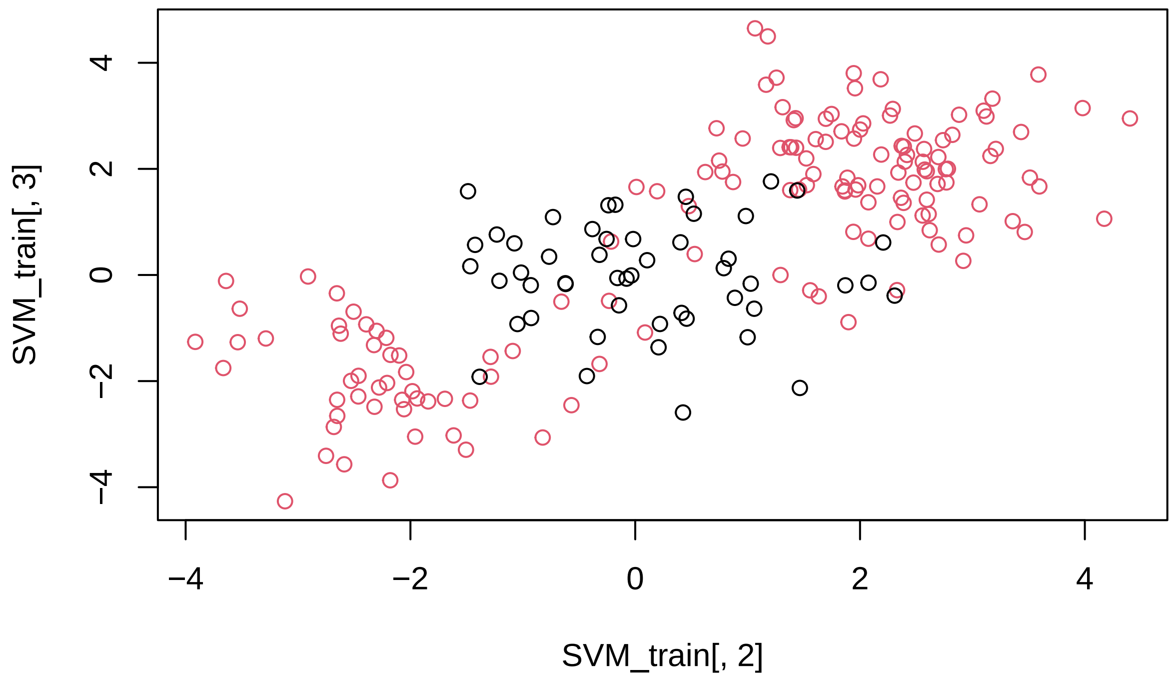
##Question 2

```
##This is a circle with radius 2 centered at (-1,2)
##For some reason, the draw.circle command in R is not working correctly for us so question 2 part a) pa
```

##Question 2 c): If the classes are defined the Red Class and the Blue Class, then (0,0) is in the blue class, (-1,1) is in the red class, and (2,2), (3,8) are in the Blue Class, The red class represents points inside our circle and the blue class represents points outside out circle. ##Question 2 d): According to slide 29 in the SVM slideshow, we know that the enlarged feature space for our equation for a circle means that X1, X1^2, X2, X2^2 are all terms of the linear decision boundary.

##Question 3

```
plot(SVM_train[,2],SVM_train[,3],col=3-SVM_train$y)
```



##So any red points are points with y=1 and all black points are points with y = 2. We see that the black points are concentrated in the center, and are separable, but not perfectly separable from the red points in the middle. We also see from this that we do not have a linear decision boundary.

##3b

```
SVM_train1=subset(SVM_train,select=-c(X))
SVM_train1$y = as.factor(SVM_train1$y)
tune.linear=tune(svm, y~.,data=SVM_train1 ,kernel ="linear",
                 ranges = list(cost=seq(0.001, 0.1, length=50)))
C = tune.linear$best.parameters$cost
C
```

```
## [1] 0.001
```

```
svmfit.linear = svm(y~., data=SVM_train1 , kernel="linear",
                    cost=C, scale=FALSE )
summary(svmfit.linear)
```

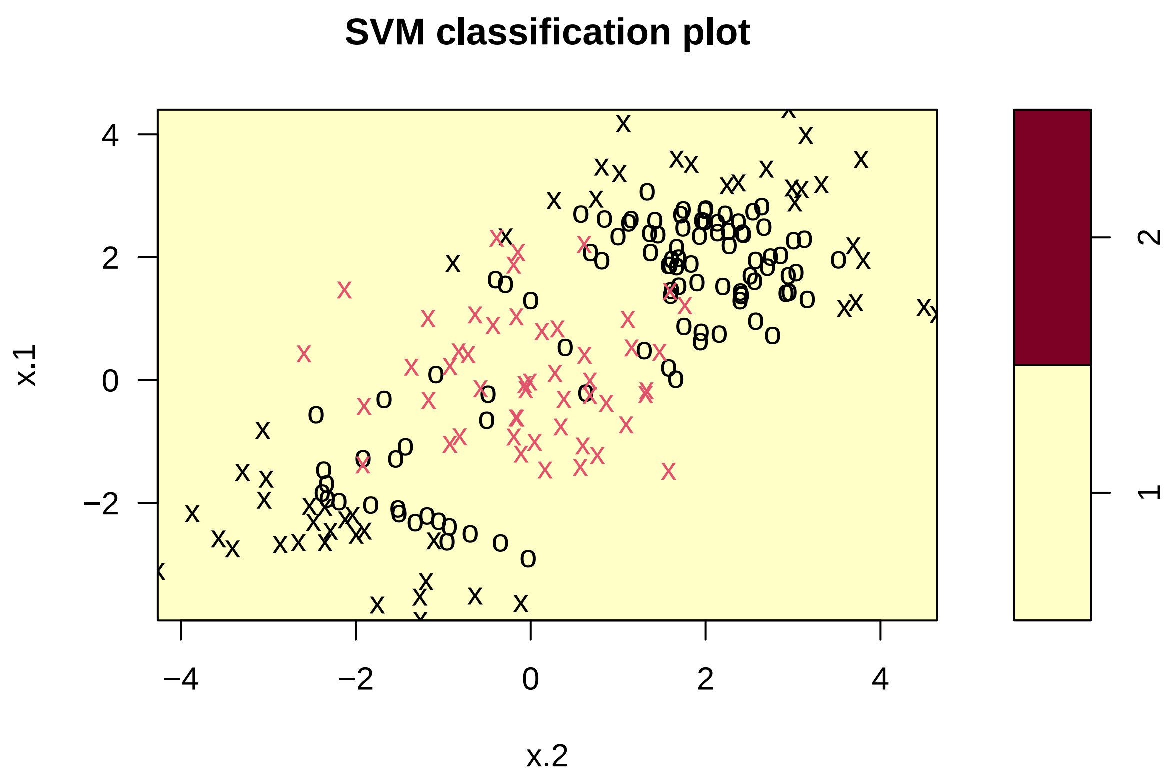
```
##
## Call:
## svm(formula = y ~ ., data = SVM_train1, kernel = "linear", cost = C,
##     scale = FALSE)
##
##
## Parameters:
```

```
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.001
##
## Number of Support Vectors:  102
##
##  ( 52 50 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2
```

```r
bestmod.linear = tune.linear$best.model
summary(bestmod.linear)
```

```
##
## Call:
## best.tune(METHOD = svm, train.x = y ~ ., data = SVM_train1, ranges = list(cost = seq(0.001,
##     0.1, length = 50)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.001
##
## Number of Support Vectors:  101
##
##  ( 51 50 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2
```

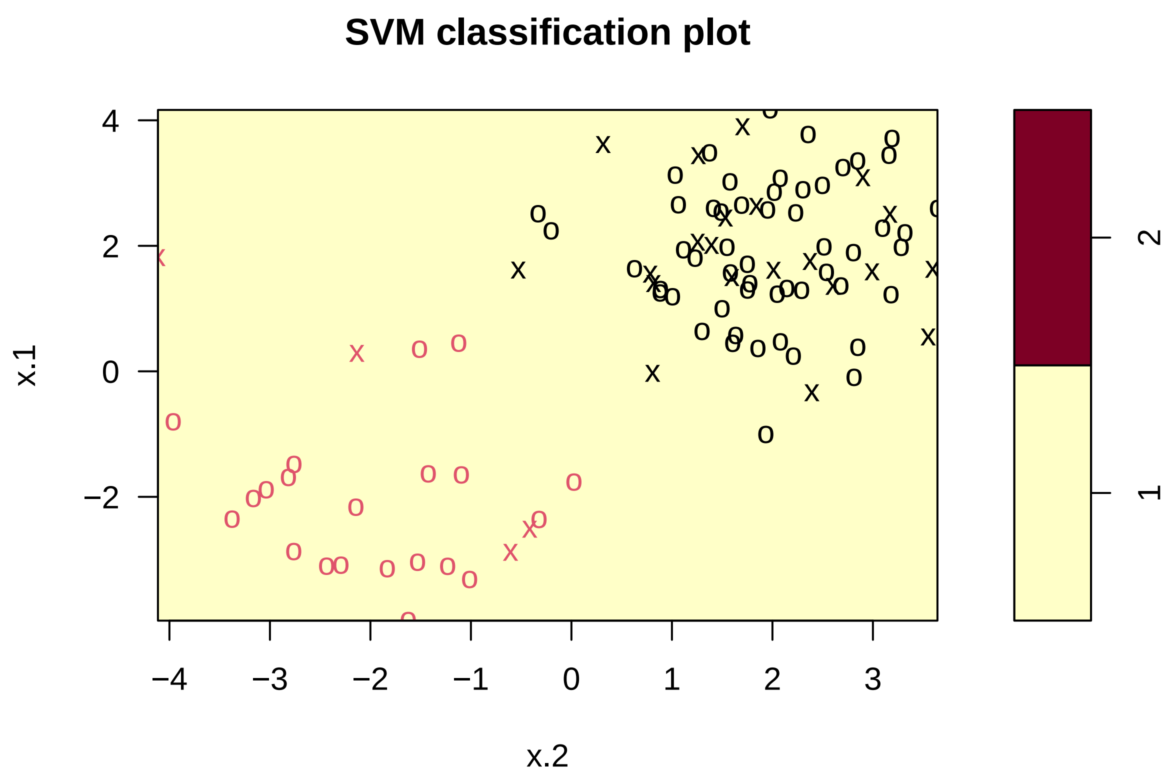
```r
plot(bestmod.linear , SVM_train1)
```




```
SVM_test1=subset(SVM_test,select=-c(X))
SVM_test1$y=as.factor(SVM_test1$y)
ypred.linear=predict(bestmod.linear, SVM_test1)
table(predict=ypred.linear , truth = SVM_test1$y )
```

```
##          truth
## predict  1  2
##        1 75 25
##        2  0  0
```

```
plot(bestmod.linear, SVM_test1)
```



##3c

```
tune.gaussian=tune(svm, y~., data=SVM_train1, kernel ="radial",
                   ranges=list(cost=seq(0.005, 0.5, length=50),
                               gamma=c(0.1,0.5,1,1.5,2)))
bestmod.gaussian =tune.gaussian$best.model
summary(bestmod.gaussian)
```

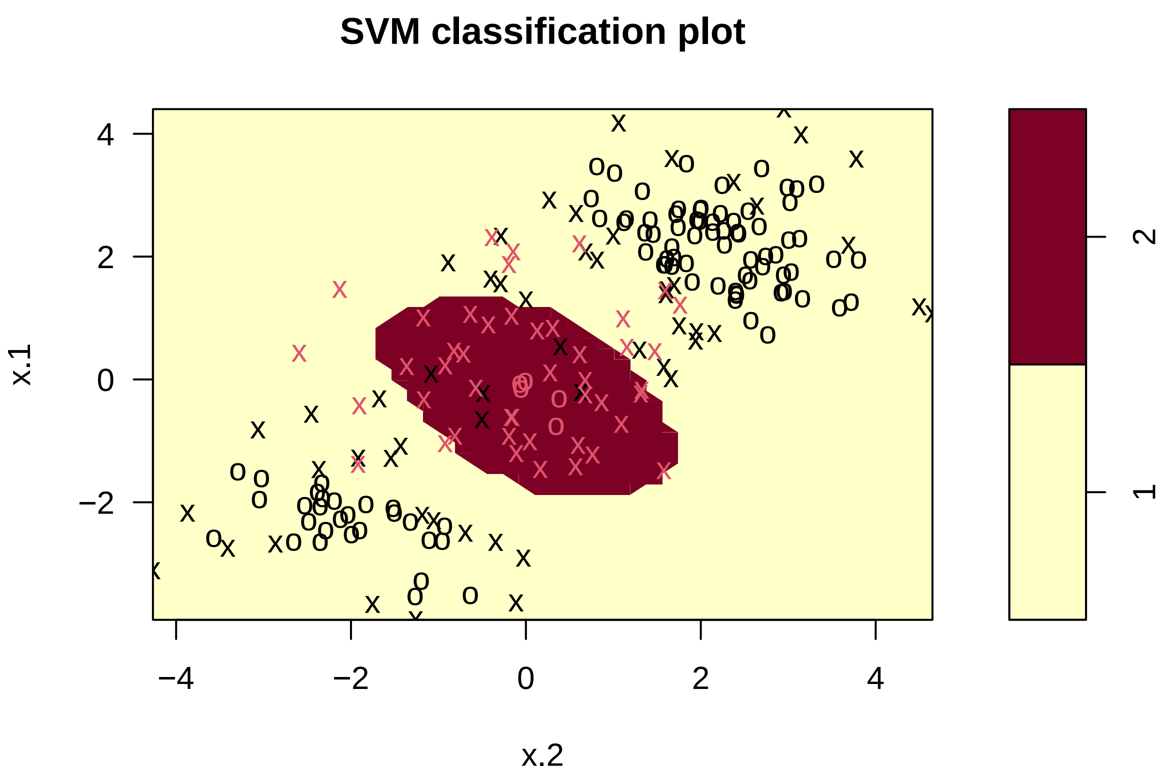
```
##
## Call:
## best.tune(METHOD = svm, train.x = y ~ ., data = SVM_train1, ranges = list(cost = seq(0.005,
##     0.5, length = 50), gamma = c(0.1, 0.5, 1, 1.5, 2)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.1363265
##
## Number of Support Vectors:  99
##
##  ( 54 45 )
##
##
## Number of Classes:  2
##
```

```
## Levels:
##  1 2
```

```
bestmod.gaussian$index
```

```
##  [1]   2   4   5   6  13  14  19  21  24  28  32  35  38  46  54  56  57  61  67
## [20]  70  74  75  77  79  84  86  91  92  94  95  99 100 103 105 106 107 108 110
## [39] 113 121 122 123 124 126 129 133 137 141 142 143 145 146 147 150 151 152 154
## [58] 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173
## [77] 175 177 178 179 180 181 182 183 184 185 187 189 190 191 192 193 194 195 196
## [96] 197 198 199 200
```

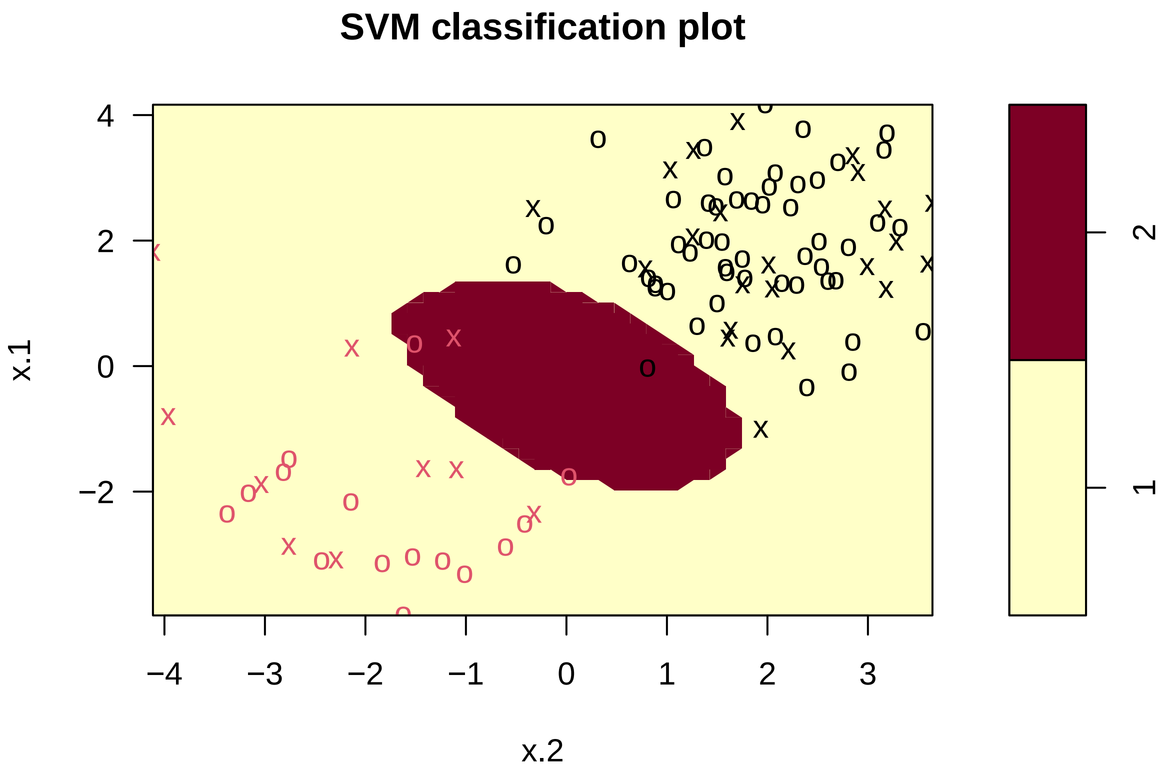
```
plot(bestmod.gaussian, SVM_train1)
```




```
ypred.gaussian=predict(bestmod.gaussian, SVM_test1)
table(predict=ypred.gaussian , truth= SVM_test1$y)
```

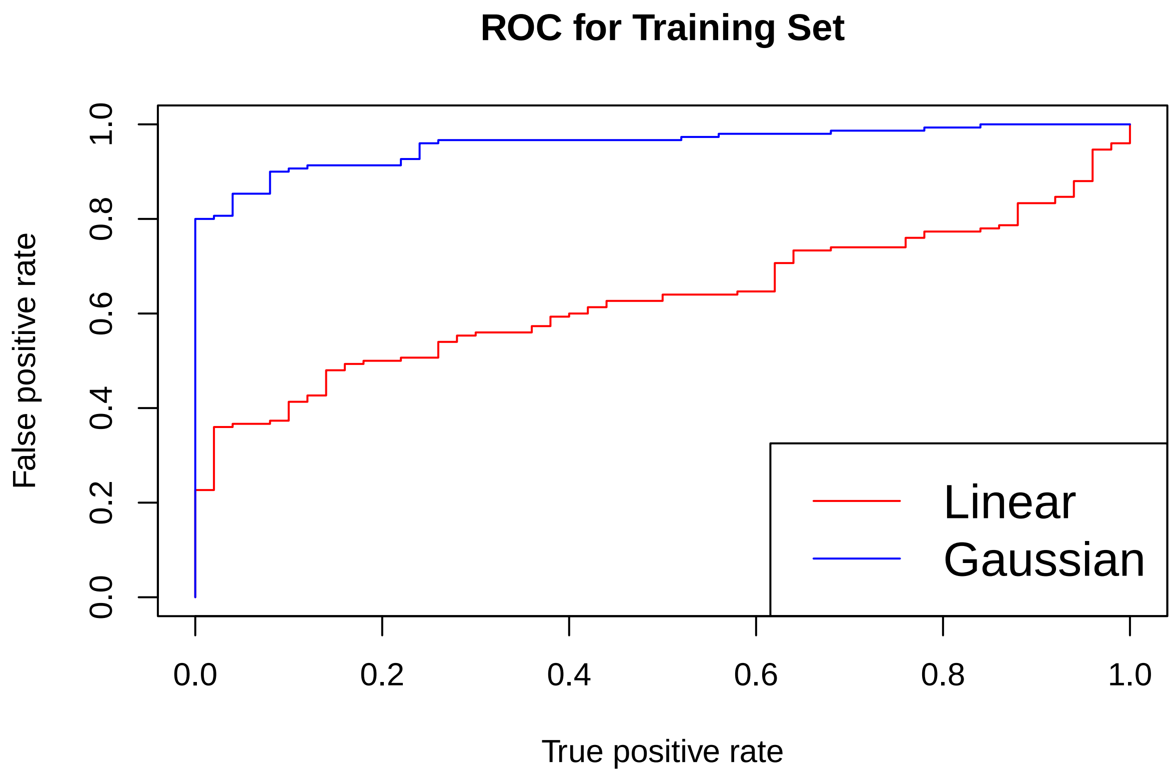
```
##        truth
## predict  1  2
##       1 74 22
##       2  1  3
```

```
plot(bestmod.gaussian,SVM_test1)
```

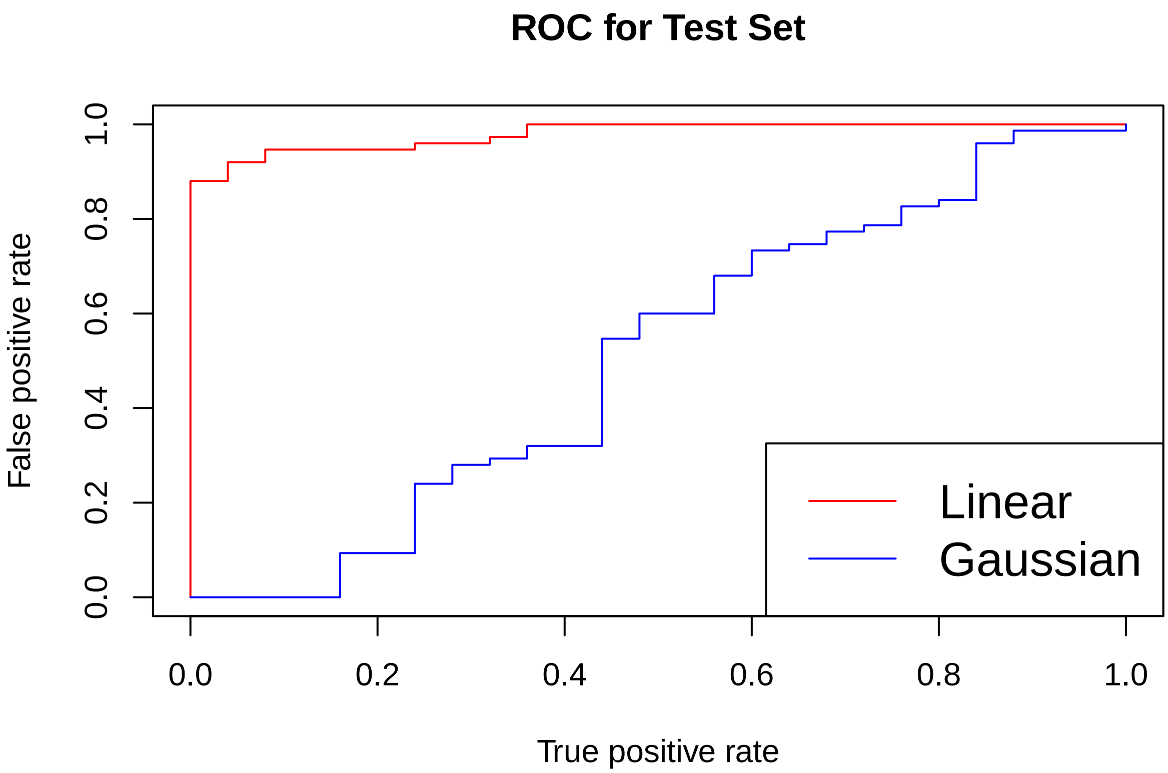


##3d

```
library(ROCR)
rocplot = function(pred, truth, ...){
  predob = prediction(pred, truth)
  perf = performance(predob, "fpr", "tpr")
  plot(perf,...)}

linear.opt=svm(y~., SVM_train1, kernel="linear", cost=C, decision.values=TRUE)
linear.train=attributes(predict(linear.opt, SVM_train1, decision.values=TRUE))$decision.values
linear.test=attributes(predict(linear.opt, SVM_test1, decision.values=TRUE))$decision.values

gaussian.opt=svm(y~., SVM_train1, kernel ="radial", gamma=tune.gaussian$best.parameters$gamma, cost=tun
gaussian.train=attributes(predict(gaussian.opt, SVM_train1, decision.values =TRUE))$decision.values
gaussian.test=attributes(predict(gaussian.opt, SVM_test1, decision.values =TRUE))$decision.values

rocplot(linear.train, SVM_train1[,"y"], main="ROC for Training Set", col="red")
rocplot(gaussian.train, SVM_train1[,"y"], add=T, col="blue")
legend("bottomright", legend=c("Linear", "Gaussian"),
       col=c("red", "blue"), lty=c(1,1), cex=1.5)
```




```
rocplot(linear.test, SVM_test1[,"y"], main="ROC for Test Set", col="red")
rocplot(gaussian.test, SVM_test1[,"y"], add=T, col="blue")
legend("bottomright", legend=c("Linear", "Gaussian"),
       col=c("red", "blue"), lty=c(1,1), cex=1.5)
```



We can clearly see that in our graph from the Training set, we see that the Gaussian model is more likely to commit false positive errors than the Linear model, and that the linear model is better than the gaussian model when applied to the training data. When we look at the Test set plot, we see that the Linear model is now more likely to commit false positive errors than the Gaussian models. This means that the Gaussian model is better on the Test data. This makes sense because we know the Gaussian model is better for predicting and generalizing.