# QDA_KNN_NB

*Jianghui Lin*

*5/15/2019*

```
test_df<-read.csv("test.csv")
train_df<-read.csv("train.csv")
```

## QDA

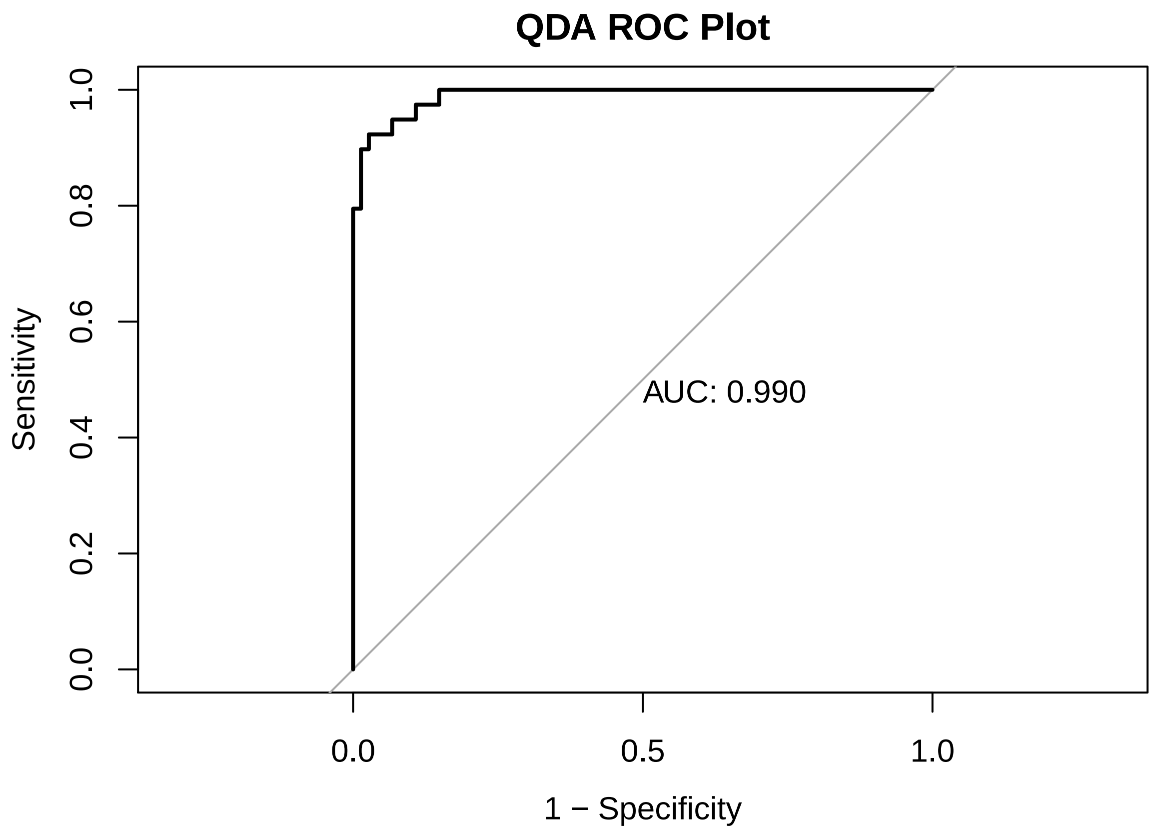
```
set.seed(1)
qda.fit <- qda(diagnosis~.,
               data = train_df)
ctrl <- trainControl(method = "repeatedcv",
                     repeats = 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
model.qda <- train(x = train_df[,-1],
                   y = train_df$diagnosis,
                   method = "qda",
                   metric = "ROC",
                   trControl = ctrl)

qda.pred <- predict(qda.fit, newdata = test_df)
head(qda.pred$posterior)
```

```
##                B            M
## 1  1.000000e+00 7.538445e-16
## 2  1.000000e+00 5.398040e-15
## 3  1.000000e+00 3.363637e-13
## 4 2.919084e-127 1.000000e+00
## 5  1.000000e+00 3.555226e-22
## 6  1.000000e+00 2.735734e-10
```

```
roc.qda <- roc(test_df$diagnosis, qda.pred$posterior[,2],
               levels = c("B","M"))

plot(roc.qda, legacy.axes = TRUE, print.auc = TRUE,main="QDA ROC Plot")
```




**AUC Value for QDA is 0.990 as shown above.**

## KNN

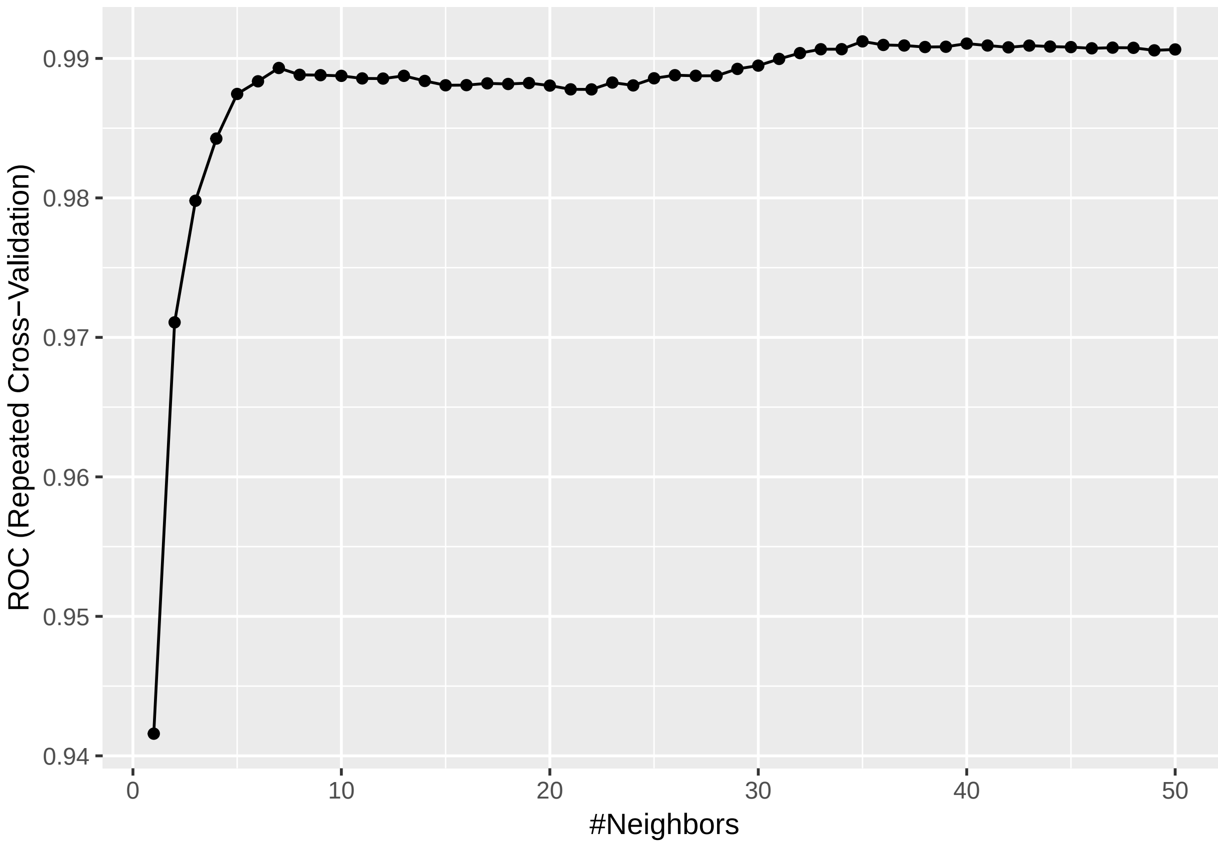
```
set.seed(1)
model.knn <- train(x = train_df[,-1],
                   y = train_df$diagnosis,
                   method = "knn",
                   preProcess = c("center", "scale"),
                   tuneGrid = data.frame(k = seq(1,50,by=1)),
                   trControl = ctrl)
```

```
## Warning in train.default(x = train_df[, -1], y = train_df$diagnosis, method
## = "knn", : The metric "Accuracy" was not in the result set. ROC will be
## used instead.
```

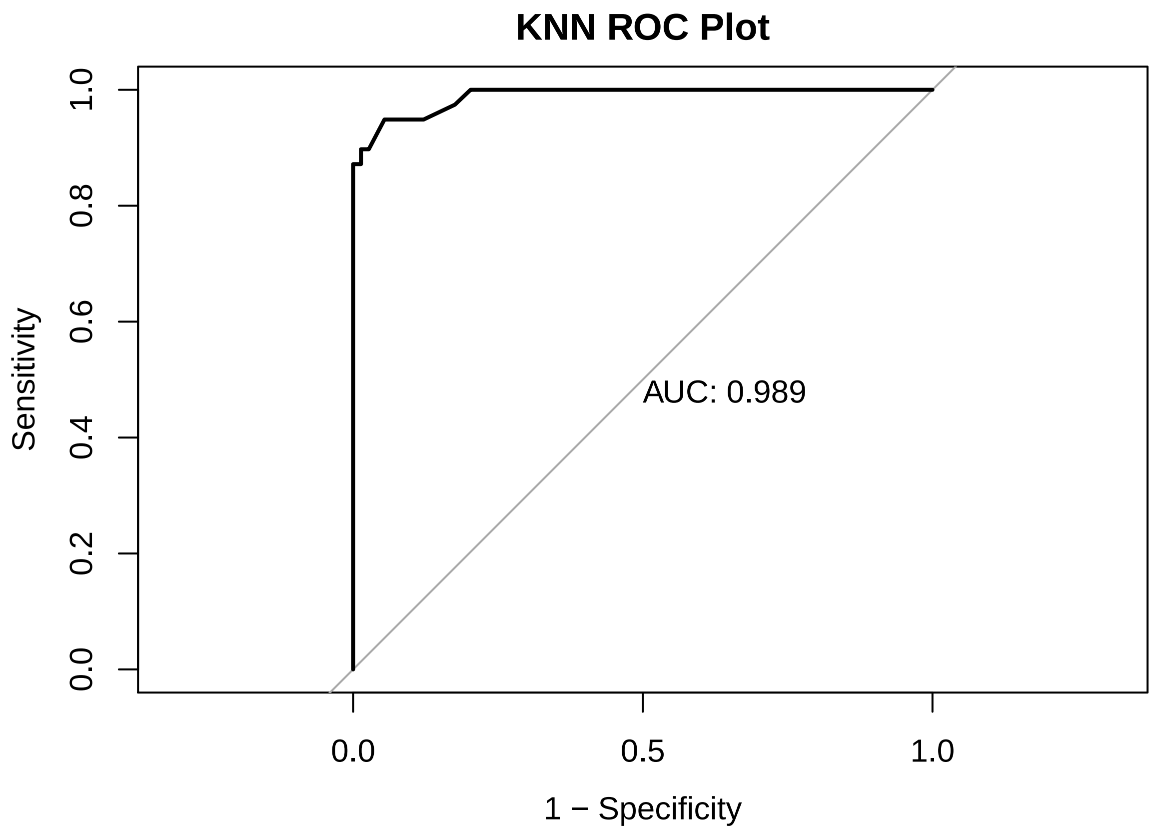
```
model.knn$bestTune
```

```
##     k
## 35 35
```

```
ggplot(model.knn)
```




```
pred_knn = predict.train(model.knn, newdata = test_df, type = 'prob')
roc_knn <- roc(test_df$diagnosis, pred_knn[,2],
               levels = c("B", "M"))
plot.roc(roc_knn, legacy.axes = TRUE, print.auc = TRUE,main="KNN ROC Plot")
```




**AUC Value for KNN is 0.989 as shown above.**

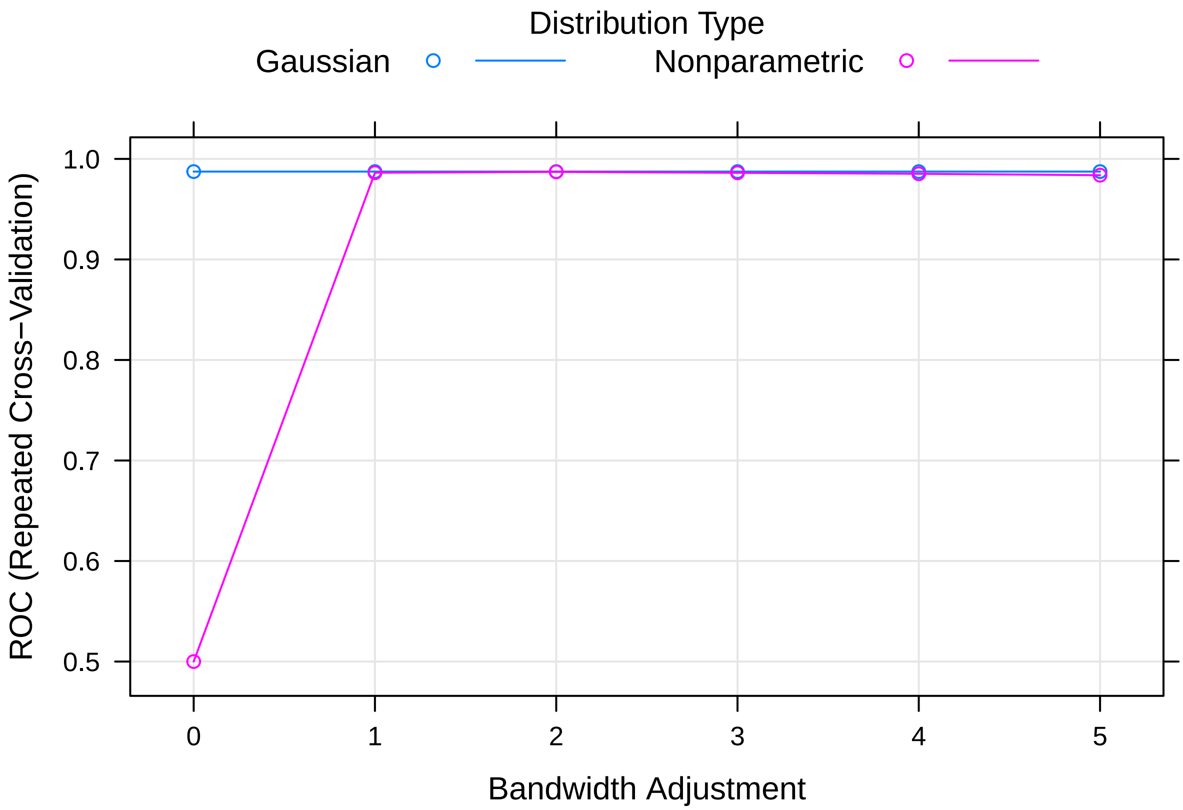
## Bayes

```
set.seed(1)

nbGrid <- expand.grid(usekernel = c(FALSE,TRUE),
                      fL = 1,
                      adjust = seq(0,5,by = 1))

model.nb <- train(x = train_df[,-1],
                  y = train_df$diagnosis,
                  method = "nb",
                  tuneGrid = nbGrid,
                  metric = "ROC",
                  trControl = ctrl)

plot(model.nb)
```




## Compare QDA, NB and KNN

```
res <- resamples(list(QDA=model.qda,NB = model.nb, KNN = model.knn))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: QDA, NB, KNN
## Number of resamples: 50
##
## ROC
##          Min.   1st Qu.    Median      Mean  3rd Qu. Max. NA's
## QDA 0.9406130 0.9879202 0.9939812 0.9911740 1.000000    1    0
## NB  0.9636015 0.9794685 0.9890008 0.9873719 0.995907    1    0
## KNN 0.9621849 0.9849138 0.9945004 0.9912221 1.000000    1    0
##
## Sens
##          Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## QDA 0.8928571 0.9642857 0.9649015 0.9682266 1.0000000    1    0
## NB  0.8571429 0.9285714 0.9642857 0.9472660 0.9655172    1    0
## KNN 0.9285714 0.9655172 1.0000000 0.9879557 1.0000000    1    0
##
## Spec
##          Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
```

```
## QDA 0.8333333 0.9411765 0.9411765 0.9513725 1.0000000    1    0
## NB  0.7058824 0.8455882 0.8888889 0.8981046 0.9411765    1    0
## KNN 0.7058824 0.8259804 0.8823529 0.8816993 0.9411765    1    0
```

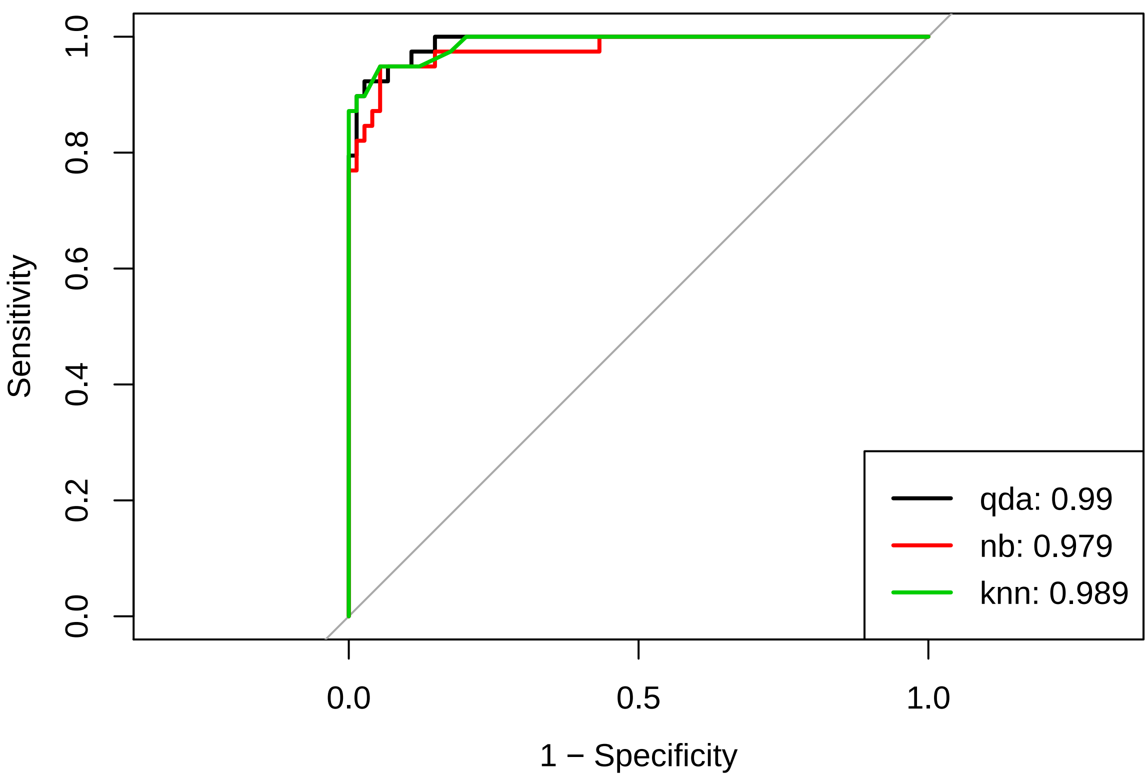
Now let's look at the test set performance.

```
library(stats)
pred_knn = predict.train(model.knn, newdata = test_df, type = 'prob')[,2]
pred_qda = predict.train(model.qda, newdata = test_df, type = 'prob')[,2]
pred_nb = predict.train(model.nb, newdata = test_df, type = 'prob')[,2]

roc.nb <- roc(test_df$diagnosis, pred_nb)
roc.qda <- roc(test_df$diagnosis, pred_qda)
roc.knn <- roc(test_df$diagnosis, pred_knn)

auc <- c(roc.qda$auc[1], roc.nb$auc[1], roc.knn$auc[1])


plot(roc.qda, col = 1,legacy.axes=TRUE)
plot(roc.nb, col = 2,add=TRUE)
plot(roc.knn, col = 3,add=TRUE)
modelNames <- c("qda","nb","knn")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc,3)),
       col = 1:6, lwd = 2)
```

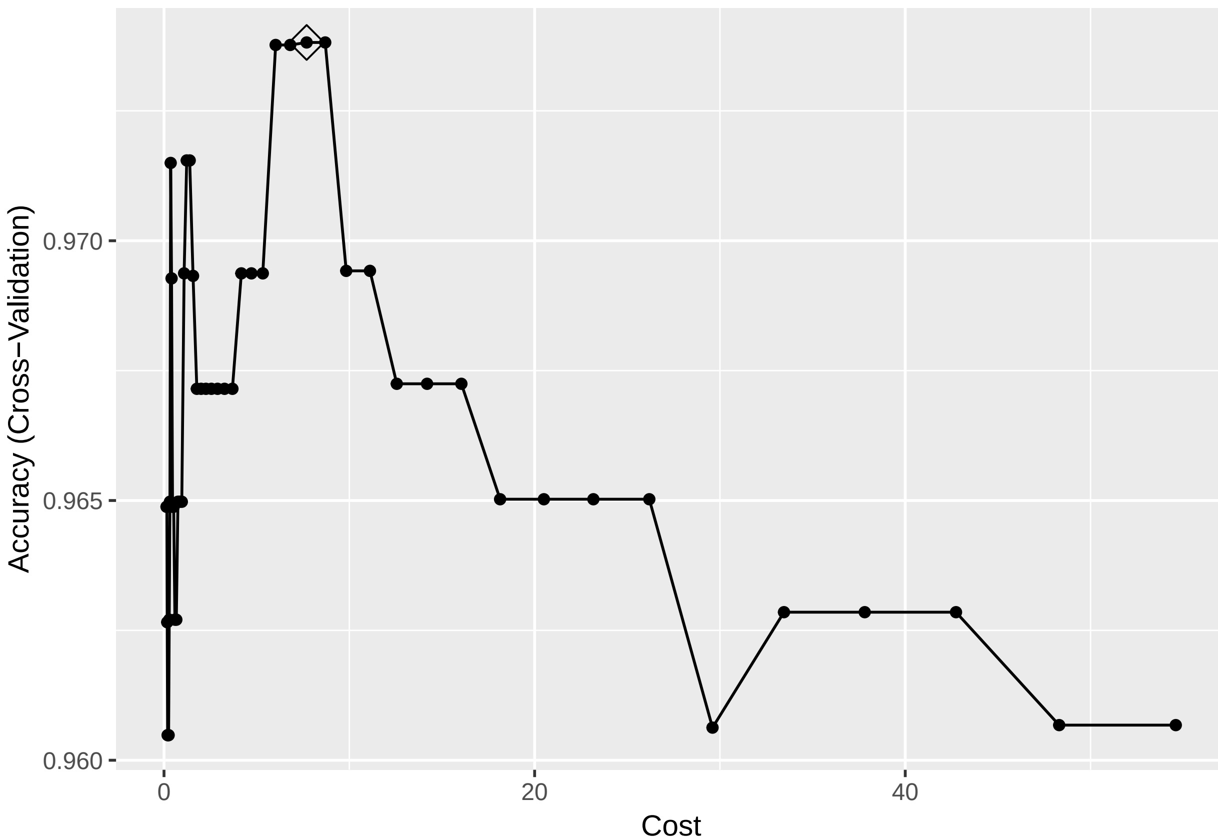



### Linear Kernel

```
##Linear Kernel
ctrl <- trainControl(method = "cv")
```

```r
set.seed(1)
svml.fit <- train(diagnosis~.,
                  data = train_df,
                  method = "svmLinear2",
                  preProcess = c("center", "scale"),
                  tuneGrid = data.frame(cost = exp(seq(-2,4,len=50))),
                  trControl = ctrl)

ggplot(svml.fit, highlight = TRUE)
```




Linear Kernel Training Error Rate

```r
pred.svml.train <- predict(svml.fit)
mean(pred.svml.train != train_df$diagnosis)
```

```
## [1] 0.00877193
```

**The training error rate for linear kernel is 0.0088.**

Linear Kernel Test Error Rate

```r
pred.svml.test <- predict(svml.fit, newdata = test_df)
mean(pred.svml.test != test_df$diagnosis)
```

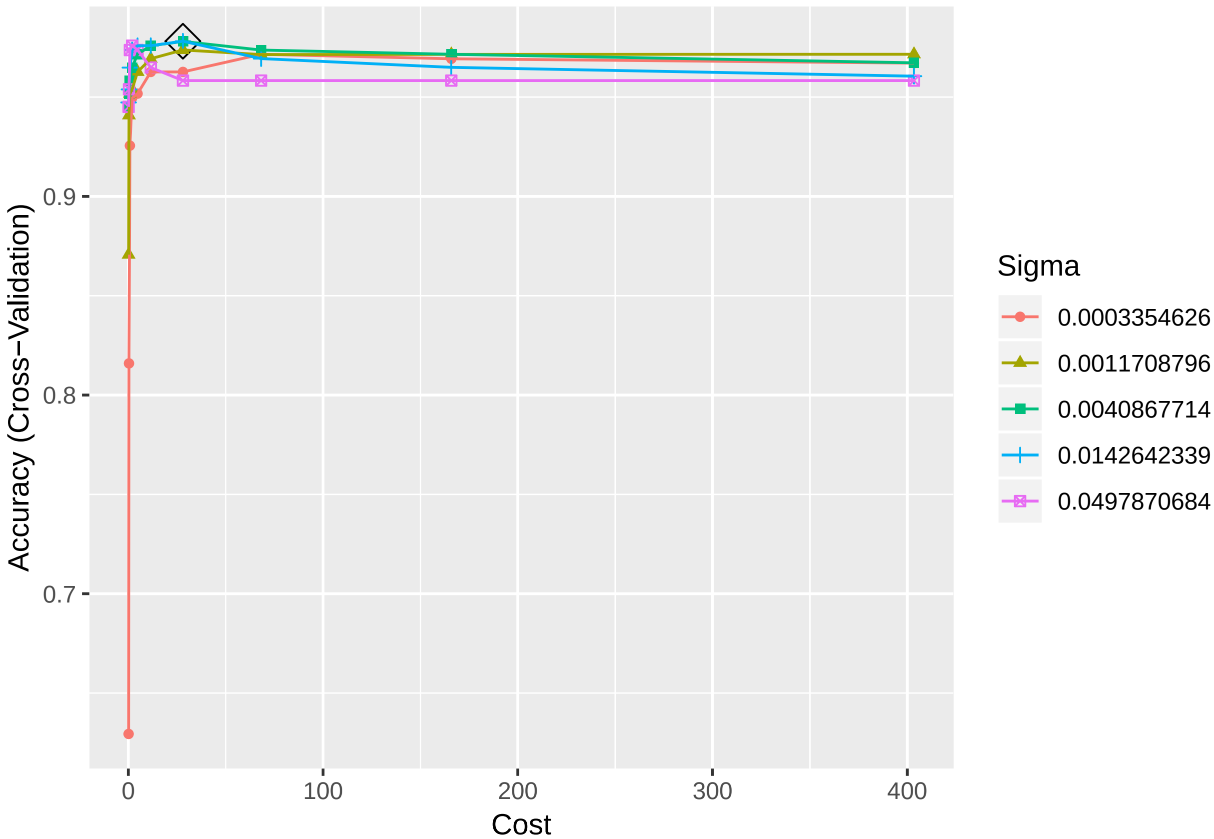
```
## [1] 0.02654867
```

**The testing error rate for linear kernel is 0.0265.**

**b)Radial Kernel** Fit a support vector machine with a radial kernel to the training data. What are thetraining and test error rates?

```r
svmr.grid <- expand.grid(C = exp(seq(-2,6,len=10)),
                         sigma = exp(seq(-8,-3,len=5)))
set.seed(1)
svmr.fit <- train(diagnosis~.,
                  data = train_df,
                  method = "svmRadial",
                  preProcess = c("center", "scale"),
                  tuneGrid = svmr.grid,
                  trControl = ctrl)

ggplot(svmr.fit, highlight = TRUE)
```




### Radial Kernel Training Error Rate

```r
pred.svmr.train <- predict(svmr.fit)
mean(pred.svmr.train != train_df$diagnosis)
```

```
## [1] 0.01096491
```

**The training error rate for radial kernel is 0.011.**

### Raidal Kernel Test Error Rate

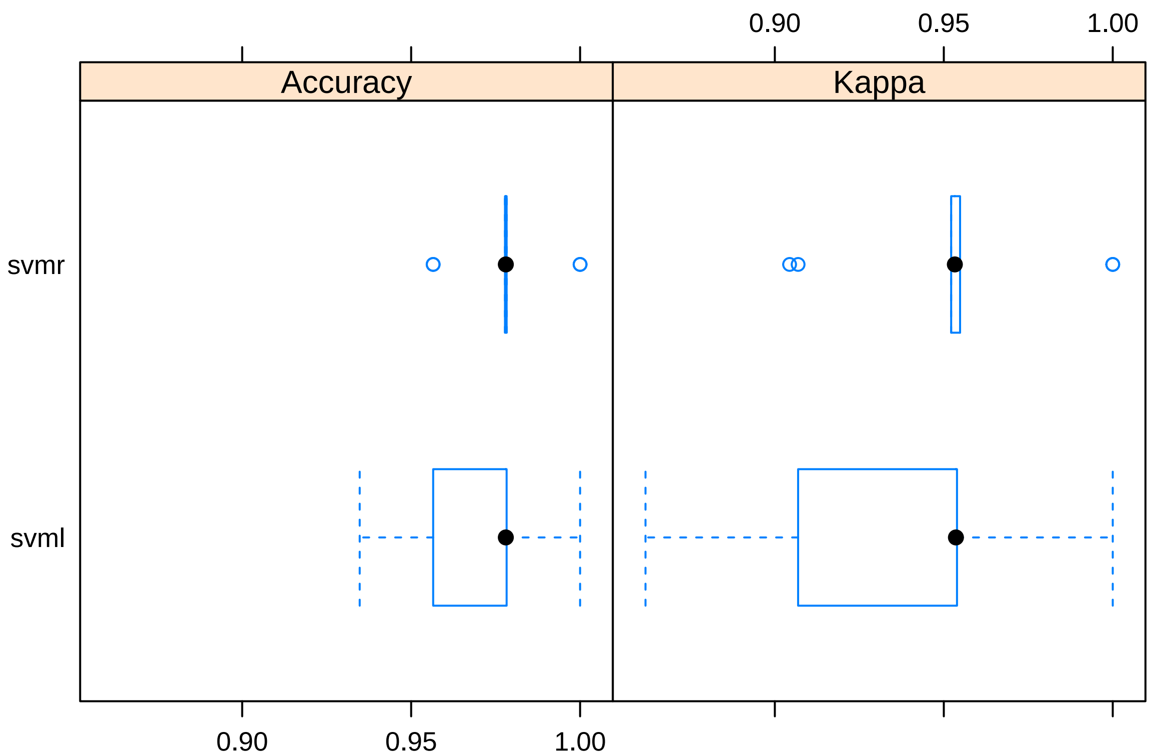
```r
pred.svmr.test <- predict(svmr.fit, newdata = test_df)
mean(pred.svmr.test != test_df$diagnosis)
```

```
## [1] 0.01769912
```

**The testing error rati for radial kernel is 0.0177.**

## (c) Which approach seems to give a better result on this data?

```r
resamp <- resamples(list(svmr = svmr.fit, svml = svml.fit))
bwplot(resamp)
```




**According to the plot, we can tell that radial kernal has higher accuracy and Kappa compared to linear kernal.**

```r
confusionMatrix(data = pred.svml.test,
                reference = test_df$diagnosis)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 73  2
##          M  1 37
##
##                Accuracy : 0.9735
##                  95% CI : (0.9244, 0.9945)
##     No Information Rate : 0.6549
##     P-Value [Acc > NIR] : <2e-16
##
```

```
##                   Kappa : 0.9409
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9865
##             Specificity : 0.9487
##          Pos Pred Value : 0.9733
##          Neg Pred Value : 0.9737
##              Prevalence : 0.6549
##          Detection Rate : 0.6460
##    Detection Prevalence : 0.6637
##       Balanced Accuracy : 0.9676
##
##        'Positive' Class : B
##
```

```r
confusionMatrix(data = pred.svmr.test,
                reference = test_df$diagnosis)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 74  2
##          M  0 37
##
##                Accuracy : 0.9823
##                  95% CI : (0.9375, 0.9978)
##     No Information Rate : 0.6549
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9604
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
##             Specificity : 0.9487
##          Pos Pred Value : 0.9737
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6549
##          Detection Rate : 0.6549
##    Detection Prevalence : 0.6726
##       Balanced Accuracy : 0.9744
##
##        'Positive' Class : B
##
```

**According to the confusion matrix,the radial kernel has higher sensitivity, specificity, PPV, NPV and Kappa compared to those of the linear kernel. In conclusion, the radial kernel seems to give a better result on the data.**

“‘