R Notebook

Code

Hide

```
#Loading and cleaning the data
attach(breast.cancer.wisconsin)

#Missing Value check and Solution
breast.cancer.wisconsin$V7 <- replace(breast.cancer.wisconsin$V7, breast.cancer.wisconsin$V7
== "?", NA)
breast.cancer.wisconsin$V7 <- as.integer(breast.cancer.wisconsin$V7)

breast_cancer_wisconsin<-na.omit(breast.cancer.wisconsin)
```

Hide

```
#Descriptive Dataset's Info
#Class Label- 2 - Benign and 4- Malignant
names(breast_cancer_wisconsin)<-c('ID', 'C_Thickness','UCSize', 'UCShape', 'M_Adhesion', 'Sin
gle_ECSize', 'Bare_Nuclei', 'Bland_Chromatin', 'Normal_Nucleoli', 'Mitoses', 'Class_Label')

#Encondings of Class Label
breast_cancer_wisconsin$Class_Label<-ifelse(breast_cancer_wisconsin$Class_Label== 2, 0, 1)
breast_cancer_wisconsin$Class_Label
```

```
  [1] 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 0 0 0 0 0 1 0
 [34] 0 0 1 0 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 1
 [67] 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1
[100] 1 0 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 0 0 0 1 0
[133] 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 1 1 0 0 1 0 0 1 1 0 0 0 0 1 1 0 0 0
[166] 0 0 1 1 1 0 1 0 1 0 0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0
[199] 0 1 1 0 0 0 1 1 0 1 1 1 0 0 1 0 0 1 1 1 1 0 1 1 0 1 1 1 0 1 0 1 1
[232] 1 1 0 0 0 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 1 0 1 0
[265] 1 1 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 1 0 1 1 0 0 1
[298] 0 0 0 1 0 0 1 1 0 0 1 0 1 0 0 1 0 1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0
[331] 1 0 0 0 1 1 0 0 0 1 0 0 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
[364] 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
[397] 0 1 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0
[430] 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
[463] 0 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
[496] 0 0 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
[529] 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 0
[562] 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 1
[595] 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0
[628] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
[661] 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1
```

Hide

```
#Divide the data into training(80%) and testing(20%)
breast_cancer_wisconsin$Class_Label <- factor(breast_cancer_wisconsin$Class_Label)

#a) Divide the data into training and testing sets
library(caret)
set.seed(1) # for reproducibility

# Create a vector of row indices
rows <- 1:nrow(breast_cancer_wisconsin)

# Randomly sample 80% of the row indices for the training set
training_rows <- sample(rows, floor(0.8 * length(rows)))

# The remaining rows are for the testing set
testing_rows <- setdiff(rows, training_rows)

training_data <- breast_cancer_wisconsin[training_rows, ]
testing_data <- breast_cancer_wisconsin[-training_rows, ]

# Create X.train and X.test data frames that exclude the class label
X.train <- training_data[, -which(names(training_data) == "Class_Label")]
X.test <- testing_data[, -which(names(testing_data) == "Class_Label")]
Y.train <- training_data$Class_Label
Y.test <- testing_data$Class_Label

#b) Division Verification in number of Examples
cat("Number of examples in training data:", nrow(training_data), "\n")
```

```
Number of examples in training data: 546
```

Hide

```
cat("Number of examples in testing data:", nrow(testing_data), "\n")
```

```
Number of examples in testing data: 137
```

Hide

```
library(e1071)
#1- svm(): build an SVM
# kernel="linear": support vector classifier
# cost argument: allows specifying the cost of a violation to the margin
# Small cost value => wider margin
# scale=FALSE: not scale each feature to have mean zero or standard deviation one
svmfit_a.linear=svm(Class_Label~., data=training_data, kernel="linear")
```

Hide

```
# Basic information about the SV classifier
summary(svmfit_a.linear)
```

```
Call:
svm(formula = Class_Label ~ ., data = training_data, kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  37

 ( 18 19 )


Number of Classes:  2

Levels:
 0 1
```

Hide

```
# Testing the model on the test set
ypred_a.linear=predict(svmfit_a.linear,testing_data)
table(predict=ypred_a.linear, truth=Y.test)
```

```
       truth
predict  0  1
      0 94  2
      1  3 38
```

Hide

```
#a) model a performance
test_a_linear_Accuracy<-mean(ypred_a.linear==Y.test)
cat("Test_a Accuracy:", test_a_linear_Accuracy, "\n")
```

```
Test_a Accuracy: 0.9635036
```

Hide

```
#b) tune(): perform cross-validation
set.seed(1)
tune.out.linear=tune(svm,Class_Label~.,data=training_data,kernel="linear",
              ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))
```

Hide

```
# CV error for each model
summary(tune.out.linear)
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
```

cost <dbl>
0.01

1 row

```
- best performance: 0.02558923

- Detailed performance results:
```

cost <dbl>	error <dbl>	dispersion <dbl>
1e-03	0.04757576	0.02867781
1e-02	0.02558923	0.02135564
1e-01	0.02740741	0.02874894
1e+00	0.02925926	0.02451701
5e+00	0.02925926	0.02597202
1e+01	0.02925926	0.02597202
1e+02	0.03296296	0.02959281

7 rows

```
NA
```

Hide

```
# Obtain the best model automatically
bestmod.linear=tune.out.linear$best.model
summary(bestmod.linear)
```

```
Call:
best.tune(METHOD = svm, train.x = Class_Label ~ ., data = training_data,
    ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)),
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01

Number of Support Vectors:  91

 ( 45 46 )


Number of Classes:  2

Levels:
 0 1
```

Hide

```
# predict(): predict the class label on a set of test observations, at any given
# value of the cost parameter
ypred_b.linear=predict(bestmod.linear, testing_data)
table(predict=ypred_b.linear, truth=Y.test)
```

```
       truth
predict  0  1
      0 94  3
      1  3 37
```

Hide

```
#model b performance
test_b_linear_Accuracy<-mean(ypred_b.linear==Y.test)
cat("Test_b Accuracy:", test_b_linear_Accuracy, "\n")
```

```
Test_b Accuracy: 0.9562044
```

Hide

```
#c)Comparison between linear a and b: Looking at model A and B performance on the test data,
both had similar accuracies on the test data, with the model using cost 1 (defaut) achieved a
slightly higher accuracy than the model using cost 0.01.
```

Hide

```
#2- svm(): build an SVM
# kernel="radial": support vector classifier
svmfit_a.radial=svm(Class_Label~., data=training_data, kernel="radial")
```

Hide

```
# Basic information about the SV classifier
summary(svmfit_a.radial)
```

```


Call:
svm(formula = Class_Label ~ ., data = training_data, kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  77

 ( 29 48 )


Number of Classes:  2

Levels:
 0 1
```

Hide

```
# Testing the model on the test set
ypred_a.radial=predict(svmfit_a.radial,testing_data)
table(predict=ypred_a.radial, truth=Y.test)
```

```
       truth
predict  0  1
      0 94  3
      1  3 37
```

Hide

```
#a) model a performance
test_a_radial_Accuracy<-mean(ypred_a.radial==Y.test)
cat("Test_a Accuracy:", test_a_radial_Accuracy, "\n")
```

```
Test_a Accuracy: 0.9562044
```

Hide

```
#b) tune(): perform cross-validation
set.seed(1)
tune.out.radial=tune(svm,Class_Label~.,data=training_data,kernel="radial",
              ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))
```

Hide

```
# CV error for each model
summary(tune.out.radial)
```

```
Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
```

cost
<dbl>
1

1 row

```
- best performance: 0.03289562

- Detailed performance results:
```

cost	error	dispersion
<dbl>	<dbl>	<dbl>
1e-03	0.36437710	0.07304989
1e-02	0.04033670	0.03435216
1e-01	0.03481481	0.03919767
1e+00	0.03289562	0.03520985
5e+00	0.04393939	0.03365179
1e+01	0.04764310	0.03247922
1e+02	0.05134680	0.03323825

7 rows

```
NA
```

Hide

```
# Obtain the best model automatically
bestmod.radial=tune.out.radial$best.model
summary(bestmod.radial)
```

```
Call:
best.tune(METHOD = svm, train.x = Class_Label ~ ., data = training_data,
    ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)),
    kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  77

 ( 29 48 )


Number of Classes:  2

Levels:
 0 1
```

Hide

```
# predict(): predict the class label on a set of test observations, at any given
# value of the cost parameter
ypred_b.radial=predict(bestmod.radial, testing_data)
table(predict=ypred_b.radial, truth=Y.test)
```

```
       truth
predict  0  1
      0 94  3
      1  3 37
```

Hide

```
#model b performance
test_b_radial_Accuracy<-mean(ypred_b.radial==Y.test)
cat("Test_b Accuracy:", test_b_radial_Accuracy, "\n")
```

```
Test_b Accuracy: 0.9562044
```

Hide

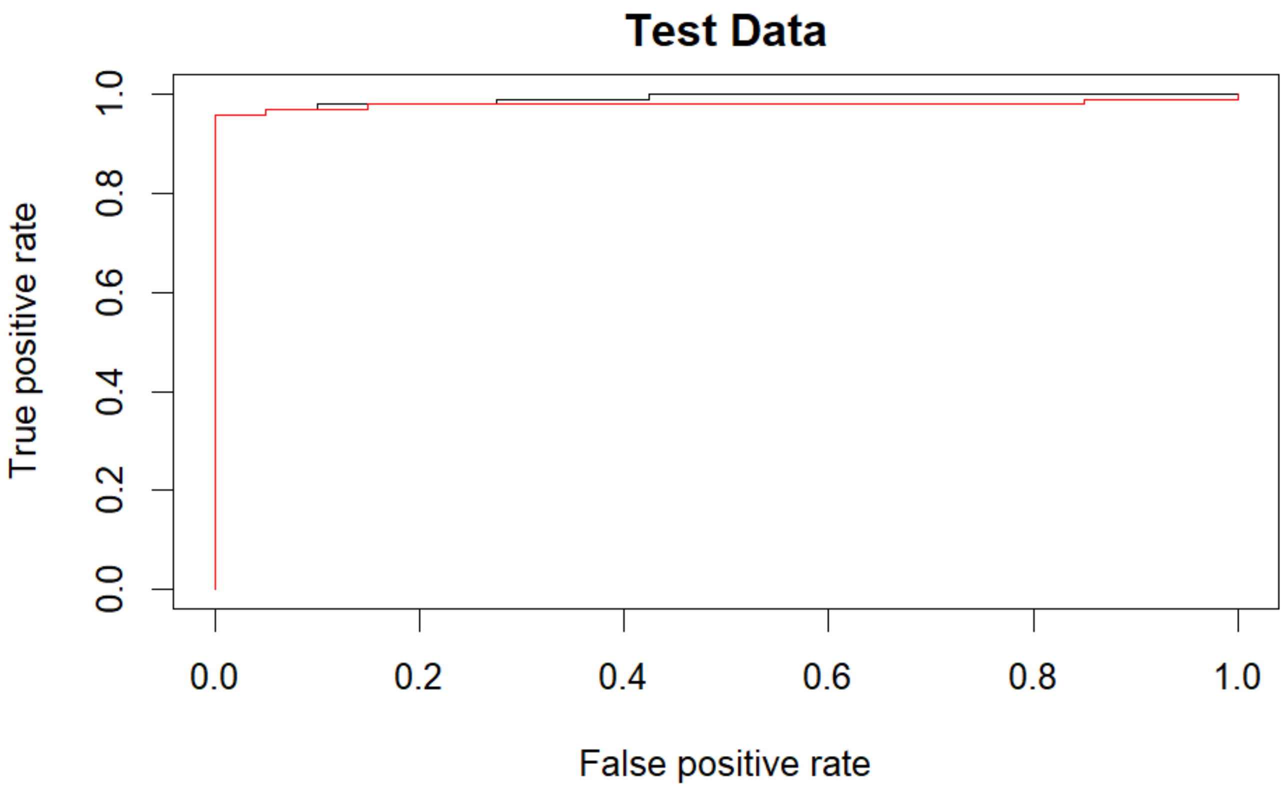
```
#c)Looking at model A and B performance on the test data, both had equal accuracies on the te
st data, with the model A using cost 1 (defaut)and  model B using cost 1.
```

Hide

```
# 3- ROC linear vs Radial
library(ROCR)

#a)Function to plot a ROC curve given a vector containing a numerical score for
# each observation, pred, and a vector containing the class label for each
# observation, truth
rocplot=function(pred, truth, ...){
  predob = prediction(pred, truth, label.ordering = c(1,0))
  perf = performance(predob , "tpr", "fpr")
  plot(perf,...)}

# Performance on test data
fitted1=attributes(predict(svmfit_a.linear,testing_data,decision.values=T))$decision.values
rocplot(fitted1,Y.test, main="Test Data")
fitted2=attributes(predict(bestmod.radial,testing_data,decision.values=T))$decision.values
rocplot(fitted2,Y.test,add=T,col="red")
```



Hide

```
#b) Comparison performance between linear and radial

#overall looking at the true positive rate, the Linear svm perform better compared to radial
kernel although radial only misclassified one less compared to linear.
```