CS-GY 6923 Machine Learning

Professor: Dr. Raman Kannan

Homework 2:
Individual Classifier

Hansheng Li

Contents

1. Import Models

Here we import the libraries and data. Then set training and valid set.

```
> library(caret)
> library(rpart)
> library(rpart.plot)
> library(nnet)
> library(e1071)
> library(class)
> library(pROC)
>
```

```
> loadData = function(csvfile) { read.csv(csvfile,head=T,sep=',',stringsAsFactors=F)} # Function to load the data
> data = loadData('Dry_Bean_Dataset.csv') # Load the data
>
> head(data)[,1:6]
   Area Perimeter MajorAxisLength MinorAxisLength AspectRation Eccentricity
1 28395   610.291        208.1781        173.8887     1.197191    0.5498122
2 28734   638.018        200.5248        182.7344     1.097356    0.4117853
3 29380   624.110        212.8261        175.9311     1.209713    0.5627273
4 30008   645.884        210.5580        182.5165     1.153638    0.4986160
5 30140   620.134        201.8479        190.2793     1.060798    0.3336797
6 30279   634.927        212.5606        181.5102     1.171067    0.5204007
>
> dim(data) # Check the number of rows and columns of the data
[1] 13611    17
> names(data) # Check the names of each column (features and class variable)
 [1] "Area"            "Perimeter"       "MajorAxisLength" "MinorAxisLength"
 [5] "AspectRation"    "Eccentricity"    "ConvexArea"      "EquivDiameter"
 [9] "Extent"          "Solidity"        "roundness"       "Compactness"
[13] "ShapeFactor1"    "ShapeFactor2"    "ShapeFactor3"    "ShapeFactor4"
[17] "Class"
> length(names(data)) # Double check that the length of "names" matches the number of columns
[1] 17
> which(names(data)=='Class') # Position of the class variable column
[1] 17
> table(data$Class)

BARBUNYA   BOMBAY     CALI DERMASON    HOROZ    SEKER     SIRA
    1322      522     1630     3546     1928     2027     2636
>
> data$Class <- as.factor(data$Class)
> set.seed(18)
> train.idx <- sample(1:nrow(data), 0.7*nrow(data),replace = F)
> data.train <- data[train.idx,]
> data.valid <- data[-train.idx,]
```

2. Multinomial Logistic Regression

Build Multinomial Logistic Regression Model

```
> log.fit <- multinom(Class ~ ., data = data.train)
# weights:  126 (102 variable)
```

```
initial  value 18538.685990
iter  10 value 13417.578057
iter  20 value 9406.201803
iter  30 value 7131.638986
iter  40 value 3444.545535
iter  50 value 2079.206940
iter  60 value 1982.080845
iter  70 value 1951.091573
iter  80 value 1925.082531
iter  90 value 1917.739223
iter 100 value 1909.562134
final  value 1909.562134
stopped after 100 iterations
```

3. Support Vector Machine Classification

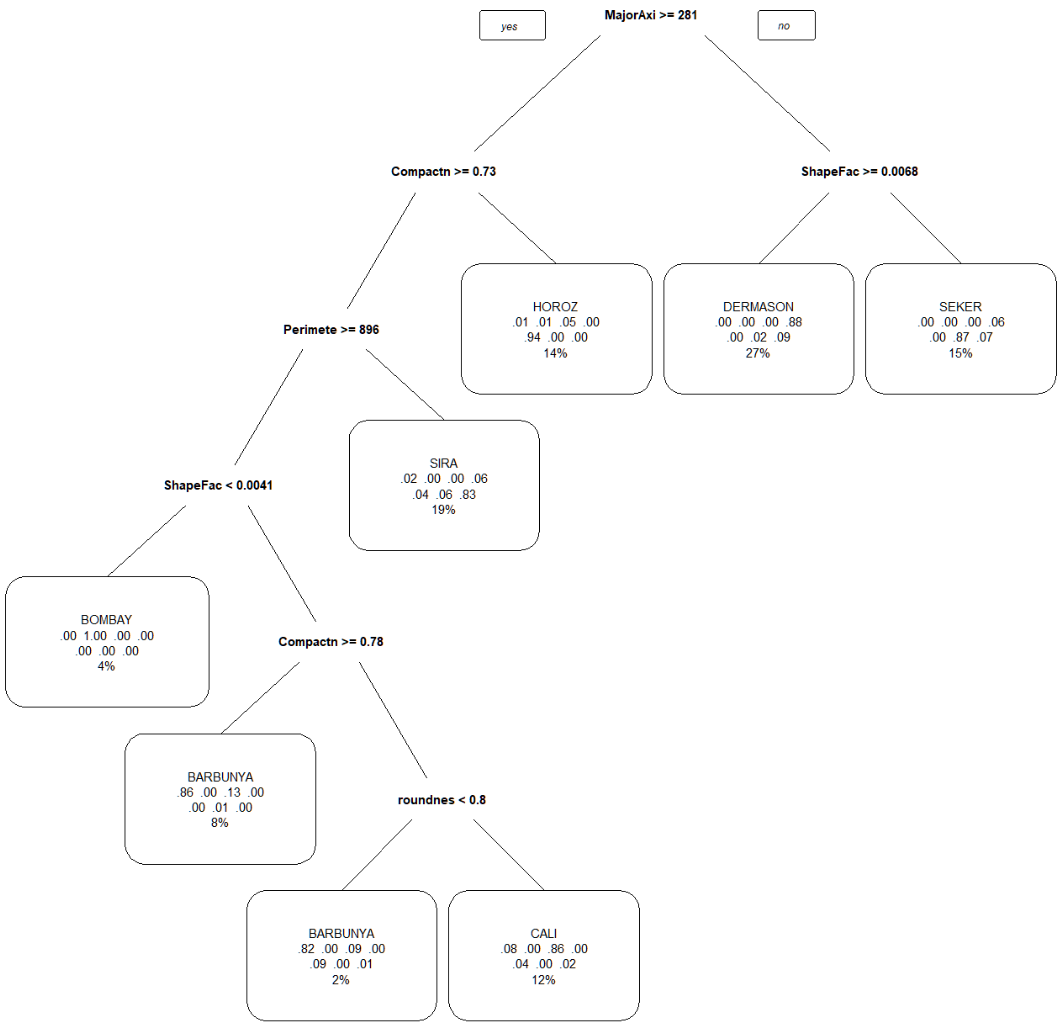
Build Support Vector Machine Model

```
> svm.fit <- svm(Class ~ ., data = data.train)
> svm.fit

Call:
svm(formula = Class ~ ., data = data.train)


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

Number of Support Vectors:  2064
```

4. Decision Tree Classification

Build Decision Tree Model

```
> tree.fit <- rpart(Class ~ ., data = data.train)
> prp(tree.fit, extra=104)
```



5. K-Nearest Neighbor Classification

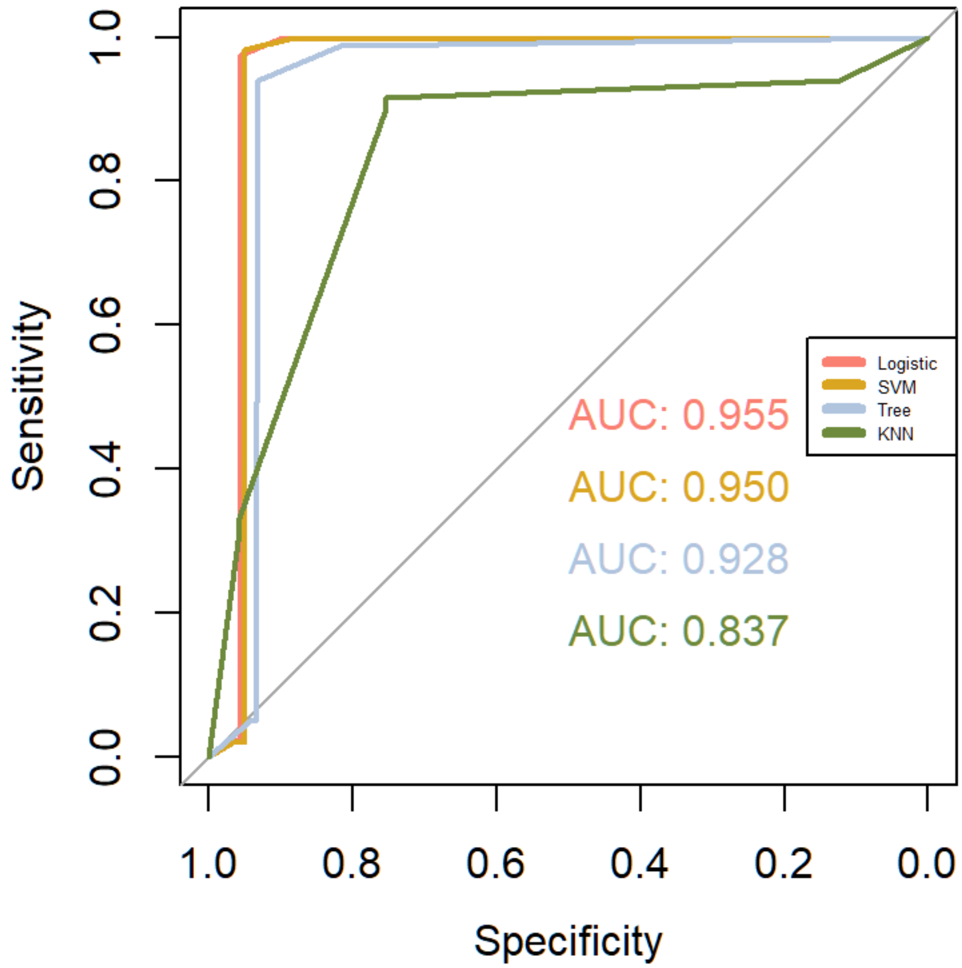
Build K-Nearest Neighbor Model. Here we set the k to square root of n.

```
> nab = as.integer(sqrt(dim(data))[1])
> knn.pred <- knn(data.train[,-c(17)],data.valid[,-
             c(17)],cl=data.train$Class,k=nab)
> knn.pred
   [1] DERMASON DERMASON DERMASON DERMASON DERMASON DERMASON DERMASON DERMAS
             ON DERMASON
  [10] DERMASON DERMASON DERMASON DERMASON DERMASON DERMASON DERMASON DERMAS
             ON DERMASON
  [19] DERMASON DERMASON DERMASON DERMASON DERMASON DERMASON DERMASON DERMAS
             ON DERMASON
  ......
```

6. ROC and AUC Performance

Build ROC and AUC diagram.

```
> data.valid$Class <- factor(data.valid$Class, ordered = T)
> log.pred <- factor(predict(log.fit, data.valid), ordered = T)
> log.roc <- multiclass.roc(data.valid$Class, log.pred,quiet=T)
> log.rs <- log.roc[['rocs']]
> svm.pred <- factor(predict(svm.fit, data.valid), ordered = T)
> svm.roc <- multiclass.roc(data.valid$Class, svm.pred)
> svm.rs <- svm.roc[['rocs']]
> tree.pred <-
    factor(predict(tree.fit, data.valid,type = "class"), ordered = T)
> tree.roc <- multiclass.roc(data.valid$Class, tree.pred)
There were 21 warnings (use warnings() to see them)
> tree.rs <- tree.roc[['rocs']]
> knn.pred <- factor(knn.pred, ordered = T)
> knn.roc <- multiclass.roc(data.valid$Class, knn.pred)
> knn.rs <- knn.roc[['rocs']]
> par(pty = "s")
> sec = 4
> plot.roc(log.rs[[sec]], col = "salmon", percent=TRUE,print.auc=T)
> plot.roc(svm.rs[[sec]], col = "goldenrod", print.auc=T, percent=TRUE,add = T, print.auc.y=.4)
> plot.roc(tree.rs[[sec]], col = "lightsteelblue", print.auc=T, percent=TRUE,add = T, print.auc.y=.3)
> plot.roc(knn.rs[[sec]], col = "darkolivegreen4", print.auc=T, percent=TRUE,add = T, print.auc.y=.2)
> legend("right",
+        legend=c("Logistic", "SVM", "Tree","KNN"),
+        col=c("salmon", "goldenrod", "lightsteelblue","darkolivegreen4"),
+        lwd=4, cex =0.4, xpd = TRUE, horiz = F)
```



Here we can see Logistic have highest AUC, SVM and decision tree were pretty close. But the AUC for KNN is not that good compared to the others.

Logistic and SVM both preformed pretty good in ROC followed by decision tree. KNN did better job at start but growth rate become slowing down after.

7. AUC Confusion Matrix

```
> (log.cM <- table(log.pred, data.valid$Class))

log.pred   BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      351      0    9        0     1     3    2
  BOMBAY          0    153    0        0     0     0    0
```

```
  CALI            21       0  456         0     13      0     0
  DERMASON         0       0    0      1019      3     11    64
  HOROZ            0       0    9         3    525      0    14
  SEKER            6       0    2        14      0    561     6
  SIRA            11       0    4        83     12     24   704
There were 50 or more warnings (use warnings() to see the first 50)
>
> (svm.cM <- table(svm.pred, data.valid$Class))

svm.pred   BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      346      0    6        0     1     1    3
  BOMBAY          0    153    0        0     0     0    0
  CALI           24      0  458        0     9     0    0
  DERMASON        0      0    0     1028     3    13   69
  HOROZ           0      0    9        3   530     0   12
  SEKER           6      0    2       14     0   558    5
  SIRA           13      0    5       74    11    27  701
>
> (tree.cM <- table(tree.pred, data.valid$Class))

tree.pred  BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      318      0   33        0     6     1    2
  BOMBAY          0    153    0        0     0     0    0
  CALI           45      0  400        0    27     0    3
  DERMASON        0      0    0     1032     2    36   98
  HOROZ           1      0   42        0   492     0    4
  SEKER           3      0    0       30     0   512   41
  SIRA           22      0    5       57    27    50  642
>
> (knn.cM <- table(knn.pred, data.valid$Class))

knn.pred   BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA       48      0   41        0    33     0    0
  BOMBAY          0    153    0        0     0     0    0
  CALI          245      0  397        0    14     0    0
  DERMASON        0      0    0      952     8   188   44
  HOROZ          80      0   39        0   315    14   66
  SEKER           0      0    0      121    11   150   73
  SIRA           16      0    3       46   173   247  607
```

8. Accuracy

Same as before, Logistic get best accuracy. KNN did not so well.

```
> log.acc <- sum(diag(log.cM)) / sum(log.cM)
> svm.acc <- sum(diag(svm.cM)) / sum(svm.cM)
```

```
> tree.acc <- sum(diag(tree.cM)) / sum(tree.cM)
> knn.acc <- sum(diag(knn.cM)) / sum(knn.cM)
> acc <- data.frame(Models = c("Logistic", "SVM", "Tree","KNN"),
+                   Accuracy = c(log.acc, svm.acc, tree.acc, knn.acc))
> acc
    Models  Accuracy
          1 Logistic 0.9228697
          2      SVM 0.9240940
          3     Tree 0.8690010
          4      KNN 0.6420176
```

9. Specificity

```
> log.conf <- confusionMatrix(log.pred, data.valid$Class)
> svm.conf <- confusionMatrix(svm.pred, data.valid$Class)
> tree.conf <- confusionMatrix(tree.pred, data.valid$Class)
> knn.conf <- confusionMatrix(knn.pred, data.valid$Class)
>
> specificity <- data.frame(cbind(log.conf$byClass[,"Specificity"],
+                                svm.conf$byClass[,"Specificity"],
+                                tree.conf$byClass[,"Specificity"],
+                                svm.conf$byClass[,"Specificity"],
+                                knn.conf$byClass[,"Specificity"]))
> names(specificity) <- c("Logistic", "SVM", "Tree","KNN")
> specificity
                  Logistic       SVM      Tree       KNN        NA
Class: BARBUNYA 0.9959405 0.9970230 0.9886333 0.9970230 0.9799729
Class: BOMBAY   1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
Class: CALI     0.9905660 0.9908435 0.9791898 0.9908435 0.9281354
Class: DERMASON 0.9736931 0.9713322 0.9541315 0.9713322 0.9190556
Class: HOROZ    0.9926346 0.9932011 0.9866856 0.9932011 0.9436261
Class: SEKER    0.9919656 0.9922525 0.9787661 0.9922525 0.9411765
Class: SIRA     0.9593200 0.9605343 0.9511233 0.9605343 0.8527626
```

10. Precision

```
> log.precision<- diag(log.cM) / rowSums(log.cM)
> svm.precision<- diag(svm.cM) / rowSums(svm.cM)
```

```
> tree.precision<- diag(tree.cM) / rowSums(tree.cM)
> knn.precision<- diag(knn.cM) / rowSums(knn.cM)
> precision <- data.frame(cbind(log.precision, svm.precision, tree.precision, knn.precision))
> precision
         log.precision svm.precision tree.precision knn.precision
BARBUNYA     0.9590164     0.9691877      0.8833333     0.3934426
BOMBAY       1.0000000     1.0000000      1.0000000     1.0000000
CALI         0.9306122     0.9327902      0.8421053     0.6051829
DERMASON     0.9288970     0.9236298      0.8835616     0.7986577
HOROZ        0.9528131     0.9566787      0.9128015     0.6128405
SEKER        0.9524618     0.9538462      0.8737201     0.4225352
SIRA         0.8400955     0.8435620      0.7995019     0.5558608
```

11. Sensitivity/Recall

```
> log.recall <- diag(log.cM) / colSums(log.cM)
> svm.recall <- diag(svm.cM) / colSums(svm.cM)
> tree.recall <- diag(tree.cM) / colSums(tree.cM)
> knn.recall <- diag(knn.cM) / colSums(knn.cM)
>
> recall <- data.frame(cbind(log.recall, svm.recall, tree.recall, knn.recall))
> recall
         log.recall svm.recall tree.recall knn.recall
BARBUNYA  0.9023136  0.8894602   0.8174807  0.1233933
BOMBAY    1.0000000  1.0000000   1.0000000  1.0000000
CALI      0.9500000  0.9541667   0.8333333  0.8270833
DERMASON  0.9106345  0.9186774   0.9222520  0.8507596
HOROZ     0.9476534  0.9566787   0.8880866  0.5685921
SEKER     0.9365609  0.9315526   0.8547579  0.2504174
SIRA      0.8911392  0.8873418   0.8126582  0.7683544
```

12. Bias

```
> log.bias <- mean(predict(log.fit, data.train) != data.train$Class)
> svm.bias <- mean(predict(svm.fit, data.train) != data.train$Class)
> tree.bias <-
          mean(predict(tree.fit, data.train, type = "class") != data.train$Class)
> bias <- data.frame(cbind(log.bias, svm.bias, tree.bias))
> bias
    log.bias   svm.bias tree.bias
1 0.07232077 0.06623281 0.1241734
```

13. Variance

```
> log.var <- mean(log.pred != data.valid$Class)
> svm.var <- mean(svm.pred != data.valid$Class)
> tree.var <- mean(as.character(tree.pred) != data.valid$Class)
> knn.var <- mean(knn.pred != data.valid$Class)
> variance <- data.frame(cbind(log.var, svm.var, tree.var, knn.var))
> variance
     log.var    svm.var tree.var   knn.var
1 0.07713026 0.07590597 0.130999 0.3579824
```

14. Compare all Model

In conclusion, logical regression and SVM preform similarly good in ROC and AUC test, decision tree is ok and KNN is not so well, so we optimal to select logical regression and SVM first. In Confusion Matrix, Accuracy, Specificity, Precision and Recall (Sensitivity), they both preform very similarly, but in Bias and Variance, we can see SVM is a little bit better, and SVM is better to prevent overfitting than logical regression, also we have small number of feature but large number of samples, in this case, I would recommend SVM over logical regression. Decision tree and KNN may be better if we use ensemble techniques or other kernel to gives more optimize, but still hard to accuracy beyond SVM.