## Read data

Data <- read.csv(file = file.choose(), header = TRUE)  
head(Data)

## Sales CompPrice Income Advertising Population Price Age Education US Urban  
## 1 9.50 138 73 11 276 120 42 17 1 1  
## 2 11.22 111 48 16 260 83 65 10 1 1  
## 3 10.06 113 35 10 269 80 59 12 1 1  
## 4 7.40 117 100 4 466 97 55 14 1 1  
## 5 4.15 141 64 3 340 128 38 13 0 1  
## 6 10.81 124 113 13 501 72 78 16 1 0  
## ShelveLoc  
## 1 1  
## 2 3  
## 3 2  
## 4 2  
## 5 1  
## 6 1

FMdat <- Data[, c(9, 6, 7)]  
head(FMdat)

## US Price Age  
## 1 1 120 42  
## 2 1 83 65  
## 3 1 80 59  
## 4 1 97 55  
## 5 0 128 38  
## 6 1 72 78

nn <- dim(FMdat)[1]  
nn

## [1] 400

test\_ind <- sample(1:nn, size = 400 \* 30 / 100, replace = FALSE)  
dtrain <- FMdat[-test\_ind, ]  
dtest <- FMdat[test\_ind, ]  
dim(dtrain)

## [1] 280 3

dim(dtest)

## [1] 120 3

ytest <- dtest$US  
  
acc\_error <- function(tab) {  
 b <- sum(tab)  
 DIM <- dim(tab)  
 if (DIM[1] == 2 && DIM[2] == 2) {  
 a = tab[1, 1] + tab[2, 2]  
 acc = a/b; error = 1 - acc  
 result <- c("Acc" = acc, "Error" = error)  
 } else {  
 a = sum(diag(tab))  
 result = c("Acc" = a / b, "Error" = 1 - a/b)  
 }  
 result  
}

# Classification ————-

## 1- SVM Model —————

library(e1071)  
  
## define Model ------------------  
set.seed(444)  
Model\_Svm <- svm(factor(US) ~ Price + Age, data = dtrain,   
kernel = "radial", gamma = 1.5,   
cost = 1, decision.values = TRUE)  
summary(Model\_Svm)

##   
## Call:  
## svm(formula = factor(US) ~ Price + Age, data = dtrain, kernel = "radial",   
## gamma = 1.5, cost = 1, decision.values = TRUE)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
##   
## Number of Support Vectors: 211  
##   
## ( 113 98 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## 0 1

pred\_LABEL\_svm <- predict(Model\_Svm, newdata = dtest)  
  
cfMat <- table(real = ytest, preds = pred\_LABEL\_svm)  
cfMat

## preds  
## real 0 1  
## 0 2 42  
## 1 4 72

res\_svm <- acc\_error(cfMat)  
res\_svm

## Acc Error   
## 0.6166667 0.3833333

pred\_SCORE\_svm <- attributes(  
 predict(Model\_Svm, dtest,   
 decision.values = TRUE) )$decision.values

## 2- Tree Model ——————-

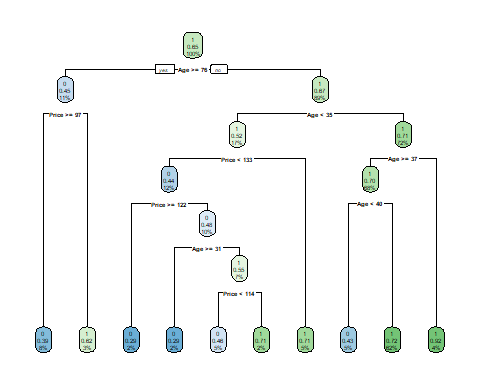
library(rpart)  
library(rpart.plot)  
Model\_Tree <- rpart(factor(US) ~ Price + Age,   
data = dtrain, parms = list(split = "information"),   
method = "class")  
  
# summary(Model\_Tree)  
##########################  
  
pred\_LABEL\_tree <- predict(Model\_Tree, dtest, type = "class")  
pred\_SCORE\_tree <- predict(Model\_Tree, dtest, type = "prob")[, 2]  
cfMat <- table(real = ytest, pred = pred\_LABEL\_tree)  
cfMat

## pred  
## real 0 1  
## 0 11 33  
## 1 18 58

res\_tree <- acc\_error(cfMat)  
res\_tree

## Acc Error   
## 0.575 0.425

rpart.plot(Model\_Tree)



## 3- Boosting Model —————

set.seed(123)  
library(gbm)  
Model\_boost <- gbm(US ~ Price + Age, data = dtrain,   
distribution = "bernoulli",   
shrinkage = 0.0001)  
pred\_SCORE\_boost <- predict(Model\_boost,   
newdata = dtest, type = "response")  
pred\_LABEL\_boost <- (pred\_SCORE\_boost > 0.5) / 1  
cfMat <- table(real = ytest,   
preds = pred\_LABEL\_boost)  
cfMat

## preds  
## real 1  
## 0 44  
## 1 76

res\_boost1 <- acc\_error(cfMat)  
res\_boost1

## Acc Error   
## 0.3666667 0.6333333

## better cutof  
  
library(ROCR)  
  
train\_pred <- predict(Model\_boost, type = "response")  
yt <- dtrain$US  
temp1 <- predict(Model\_boost, type = "response")  
temp2 <- prediction(temp1, yt)   
cost\_perf = performance(temp2, "cost")  
Cut\_p <- cost\_perf@x.values[[1]][which.min(cost\_perf@y.values[[1]])]   
Cut\_p

## [1] 0.6494601

pred\_LABEL\_boost2 <- (pred\_SCORE\_boost > Cut\_p) / 1   
tab <- table(real = ytest, pred = pred\_LABEL\_boost2)  
tab

## pred  
## real 0 1  
## 0 4 40  
## 1 7 69

res\_boost\_2 <- acc\_error(tab)  
res\_boost\_2

## Acc Error   
## 0.6083333 0.3916667

## 4- Logistic Model —————–

set.seed(123)  
Model\_log <- glm(US ~ Price + Age, data = dtrain,  
family = binomial)  
pred\_SCORE\_log <- predict(Model\_log, dtest,   
type = "response")  
pred\_LABEL\_log <- (pred\_SCORE\_log > 0.5) / 1  
  
tab <- table(real = ytest, pred = pred\_LABEL\_log)   
tab

## pred  
## real 1  
## 0 44  
## 1 76

res\_log <- acc\_error(tab)  
res\_log

## Acc Error   
## 0.3666667 0.6333333

## change cutpoint   
  
  
temp1 <- predict(Model\_log, type = "response")  
temp2 <- prediction(temp1, dtrain$US)   
cost\_perf = performance(temp2, "cost")  
Cut\_p <- cost\_perf@x.values[[1]][which.min(cost\_perf@y.values[[1]])]   
Cut\_p

## 172   
## 0.593401

pred\_LABEL\_log2 <- (pred\_SCORE\_log > Cut\_p) / 1   
tab <- table(real = ytest, pred = pred\_LABEL\_log2)  
tab

## pred  
## real 1  
## 0 44  
## 1 76

res\_log\_2 <- acc\_error(tab)  
res\_log\_2

## Acc Error   
## 0.3666667 0.6333333

## 5- QDA Model ————————

library(MASS)  
Model\_qda <- qda(US ~ Price + Age, data = dtrain)  
pred <- predict(Model\_qda, newdata = dtest,   
type = "class")  
  
pred\_LABEL\_qda <- pred$class  
pred\_SCORE\_qda <- pred$posterior[, 2]  
tab <- table(real = ytest, pred = pred\_LABEL\_qda)  
tab

## pred  
## real 0 1  
## 0 4 40  
## 1 2 74

res\_qda <- acc\_error(tab)  
res\_qda

## Acc Error   
## 0.65 0.35

\*\*\*  
\*\*\*  
  
  
## 6- RandomForest Model -----------  
  
  
  
```r  
set.seed(111)  
library(randomForest)  
  
Model\_rf <- randomForest(factor(US) ~ Price + Age,   
data = dtrain,   
ntree = 300)  
pred\_LABEL\_rf <- predict(Model\_rf,   
dtest, type = "response")  
pred\_SCORE\_rf <- predict(Model\_rf,   
dtest, type = "prob")[, 2]  
  
tab <- table(real = ytest,   
preds = pred\_LABEL\_rf)  
tab

## preds  
## real 0 1  
## 0 14 30  
## 1 22 54

res\_rf <- acc\_error(tab)  
res\_rf

## Acc Error   
## 0.5666667 0.4333333

## 7- LDA Model ——————-

library(MASS)  
Model\_lda <- lda(US ~ Price + Age, data = dtrain)  
pred <- predict(Model\_lda, newdata = dtest, type = "class")  
pred\_LABEL\_lda <- pred$class  
pred\_SCORE\_lda <- pred$posterior[, 2]  
tab <- table(pred\_LABEL\_lda, ytest)  
tab

## ytest  
## pred\_LABEL\_lda 0 1  
## 0 0 0  
## 1 44 76

res\_lda <- acc\_error(tab)  
  
res\_lda

## Acc Error   
## 0.6333333 0.3666667

## 8- bagging ——————-

library(caret)  
cvcontrol <- trainControl(method="repeatedcv", number = 5,  
 allowParallel=TRUE)  
Model\_bag <- train(factor(US) ~ Price + Age, data = dtrain,   
method = "treebag",   
trControl = cvcontrol,   
importance = TRUE)   
  
Model\_bag

## Bagged CART   
##   
## 280 samples  
## 2 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold, repeated 1 times)   
## Summary of sample sizes: 223, 224, 224, 225, 224   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.5681852 0.01689916

pred\_SCORE\_bag <- predict(Model\_bag, dtest,   
type = "prob")[, '1']  
  
pred\_LABEL\_bag <- predict(Model\_bag, dtest, type = "raw")  
tab <- table(real = ytest, pred = pred\_LABEL\_bag)  
tab

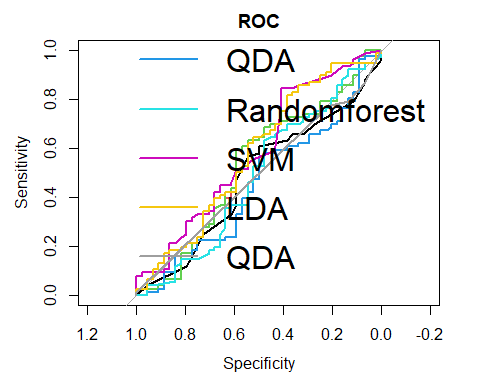
## pred  
## real 0 1  
## 0 16 28  
## 1 26 50

res\_bag <- acc\_error(tab)  
res\_bag

## Acc Error   
## 0.55 0.45

## Results —————

library(pROC)  
roc\_bag <- roc(ytest, pred\_SCORE\_bag)  
roc\_log <- roc(ytest, pred\_SCORE\_log)  
roc\_lda <- roc(ytest, pred\_SCORE\_lda)  
roc\_qda <- roc(ytest, pred\_SCORE\_qda)  
roc\_rf <- roc(ytest, pred\_SCORE\_rf)  
roc\_boost <- roc(ytest, pred\_SCORE\_boost)  
roc\_svm <- roc(ytest, pred\_SCORE\_svm)  
roc\_tree <- roc(ytest, pred\_SCORE\_tree)  
plot(roc\_bag, main = "ROC", col = 1)  
plot(roc\_log, col = 2, add = T)  
plot(roc\_lda, col = 3, add = T)  
plot(roc\_qda, col = 4, add = T)  
plot(roc\_rf, col = 5, add = T)  
plot(roc\_boost, col = 6, add = T)  
plot(roc\_svm, col = 7, add = T)  
plot(roc\_tree, col = 8, add = T)  
legend("bottomright",   
legend = c("Bagging", "Logistic", "LDA",   
"QDA", "Randomforest", "SVM", "LDA", "QDA"),   
col = 1:8, lwd = 2, lty = 1, bty = "n", cex = 2)



Models <- c("bagging", "logistic", "LDA", "QDA",   
"Randomforest", "Boosting", "SVM", "TREE")  
  
AUC <- c(auc(roc\_bag), auc(roc\_log), auc(roc\_lda),   
auc(roc\_qda), auc(roc\_rf), auc(roc\_boost), auc(roc\_svm),   
auc(roc\_tree))  
  
ERROR = c(res\_bag[2], res\_log\_2[2], res\_lda[2],   
res\_qda[2], res\_rf[2], res\_boost\_2[2], res\_svm[2], res\_tree[2])  
  
data\_result <- data.frame(Model = Models,   
AUC = AUC, Error = ERROR)  
  
knitr :: kable(data\_result, caption = "RESULTS", align = "c")

RESULTS

| Model | AUC | Error |
| --- | --- | --- |
| bagging | 0.4807117 | 0.4500000 |
| logistic | 0.5297548 | 0.6333333 |
| LDA | 0.5303529 | 0.3666667 |
| QDA | 0.4502093 | 0.3500000 |
| Randomforest | 0.4848983 | 0.4333333 |
| Boosting | 0.5823864 | 0.3916667 |
| SVM | 0.5746112 | 0.3833333 |
| TREE | 0.4914773 | 0.4250000 |

# clustering —————

library(ISLR)  
names(USArrests)

## [1] "Murder" "Assault" "UrbanPop" "Rape"

dd <- USArrests[c('Assault', "UrbanPop")]  
dd2 <- scale(dd)  
tail(dd2, 4)

## Assault UrbanPop  
## Washington -0.3091039 0.51537975  
## West Virginia -1.0770641 -1.83353601  
## Wisconsin -1.4130466 0.03177945  
## Wyoming -0.1171139 -0.38273510

distt <- dist(dd2, method = "euclidean")  
  
average\_model <- hclust(distt, method = "average")  
  
plot(average\_model, hang = -1,   
main = "Dendogram Average Linkage")

