R Programs for Text Book Examples

Chapter 9: Classification and Regression Trees

Code for running and plotting classification tree with single split. Tree representation of first split (Figure 9.7)

```
####
library(rpart)
library(rpart.plot)
mower.df <- read.csv("RidingMowers.csv")

# use rpart() to run a classification tree.
# define rpart.control() in rpart() to determine the depth of the tree.
class.tree <- rpart(Ownership ~ ., data = mower.df,
                    control = rpart.control(maxdepth = 2), method = "class")
## plot tree
# use prp() to plot the tree. You can control plotting parameters such as color, shape,
# and information displayed (which and where).
prp(class.tree, type = 1, extra = 1, split.font = 1, varlen = -10)
```

Code for creating a default classification tree. Default classification tree for the loan acceptance data using the training set (3000 records) (Figure 9.10)

```
####
library(rpart)
library(rpart.plot)

bank.df <- read.csv("UniversalBank.csv")
bank.df <- bank.df[ , -c(1, 5)]  # Drop ID and zip code columns.

# partition
set.seed(1)
train.index <- sample(c(1:dim(bank.df)[1]), dim(bank.df)[1]*0.6)
train.df <- bank.df[train.index, ]
valid.df <- bank.df[-train.index, ]

# classification tree
default.ct <- rpart(Personal.Loan ~ ., data = train.df, method = "class")
# plot tree
prp(default.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10)
```

Code for creating a deeper classification tree. A full tree for the loan acceptance data using the training set (3000 records) (Figure 9.10)

```
####
```

```
deeper.ct <- rpart(Personal.Loan ~ ., data = train.df, method = "class", cp = 0, minsplit = 1)
# count number of leaves
length(deeper.ct$frame$var[deeper.ct$frame$var == "<leaf>"])
# plot tree
prp(deeper.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10,
   box.col=ifelse(deeper.ct$frame$var == "<leaf>", 'gray', 'white'))
```

Confusion matrices and accuracy for the default (small) and deeper (full) classification trees, on the training and validation sets of the personal loan data (Figure 9.11)

```
####
# classify records in the validation data.
# set argument type = "class" in predict() to generate predicted class membership.
default.ct.point.pred.train <- predict(default.ct,train.df,type = "class")
# generate confusion matrix for training data
confusionMatrix(default.ct.point.pred.train, train.df$Personal.Loan)
### repeat the code for the validation set, and the deeper tree
```

Table of complexity parameter (CP) values and associated tree errors (Figure 9.13)

```
####
# argument xval refers to the number of folds to use in rpart's built-in
# cross-validation procedure
# argument cp sets the smallest value for the complexity parameter.
cv.ct <- rpart(Personal.Loan ~ ., data = train.df, method = "class",
          cp = 0.00001, minsplit = 5, xval = 5)
# use printcp() to print the table.
printcp(cv.ct)
```

Code for pruning the tree. Pruned classification tree for the loan acceptance data using CP that yielded lowest error (Figure 9.15))

```
####
# prune by lower cp
pruned.ct <- prune(cv.ct,
            cp = cv.ct$cptable[which.min(cv.ct$cptable[,"xerror"]),"CP"])
length(pruned.ct$frame$var[pruned.ct$frame$var == "<leaf>"])
prp(pruned.ct, type = 1, extra = 1, split.font = 1, varlen = -10)
```

Code for running a random forest, plotting variable importance plot, and computing accuracy. Variable importance plot from Random forest (Personal Loan Example) (Section 9.9))

```
####
library(randomForest)
## random forest
rf <- randomForest(as.factor(Personal.Loan) ~ ., data = train.df, ntree = 500,
            mtry = 4, nodesize = 5, importance = TRUE)
```

```
## variable importance plot
varImpPlot(rf, type = 1)

## confusion matrix
rf.pred <- predict(rf, valid.df)
confusionMatrix(rf.pred, valid.df$Personal.Loan)
```

Boosted tree: confusion matrix for the validation set (loan data) (Section 9.9)

####

```
library(adabag)
library(rpart)
library(caret)

train.df$Personal.Loan <- as.factor(train.df$Personal.Loan)

set.seed(1)
boost <- boosting(Personal.Loan ~ ., data = train.df)
pred <- predict(boost, valid.df)
confusionMatrix(pred$class, valid.df$Personal.Loan)
```

Chapter 10: Logistic Regression

Code for fitting a logistic regression model. Logistic regression model for loan acceptance (training data) (Figure 10.3)

####

```
bank.df <- read.csv("UniversalBank.csv")
bank.df <- bank.df[ , -c(1, 5)]  # Drop ID and zip code columns.
# treat Education as categorical (R will create dummy variables)
bank.df$Education <- factor(bank.df$Education, levels = c(1, 2, 3), labels = c("Undergrad", "Graduate",
"Advanced/Professional"))

# partition data
set.seed(2)
train.index <- sample(c(1:dim(bank.df)[1]), dim(bank.df)[1]*0.6)
train.df <- bank.df[train.index, ]
valid.df <- bank.df[-train.index, ]

# run logistic regression
# use glm() (general linear model) with family = "binomial" to fit a logistic
# regression.
logit.reg <- glm(Personal.Loan ~ ., data = train.df, family = "binomial")
options(scipen=999)
summary(logit.reg)
```

Code for using logistic regression to generate predicted probabilities. Propensities for the first five customers in validation data (Figure 10.4))

```
####
# use predict() with type = "response" to compute predicted probabilities.
logit.reg.pred <- predict(logit.reg, valid.df[, -8], type = "response")

# first 5 actual and predicted records
data.frame(actual = valid.df$Personal.Loan[1:5], predicted = logit.reg.pred[1:5])
```

Code for creating lift chart and decile-wise lift chart. Lift chart and decile-wise lift chart for the validation data for Universal Bank loan offer. (Figure 10.5))

```
####
library(gains)
gain <- gains(valid.df$Personal.Loan, logit.reg.pred, groups=10)

# plot lift chart
plot(c(0,gain$cume.pct.of.total*sum(valid.df$Personal.Loan))~c(0,gain$cume.obs), xlab="# cases",
ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.df$Personal.Loan))~c(0, dim(valid.df)[1]), lty=2)

# compute deciles and plot decile-wise chart
heights <- gain$mean.resp/mean(valid.df$Personal.Loan)
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9), xlab = "Percentile", ylab = "Mean
Response", main = "Decile-wise lift chart")

# add labels to columns
text(midpoints, heights+0.5, labels=round(heights, 1), cex = 0.8)
```

Code for generating bar charts of average delay vs. predictors. Proportion of delayed flights by each of the six predictors. Time of day is divided into hourly bins (Figure 10.6)

```
####
# code for generating top-right bar chart
# for other plots, replace aggregating variable by setting argument by =  in
# aggregate().
# in function barplot(), set the x-label (argument xlab =) and y-label
# (argument names.arg =)
# according to the variable of choice.

barplot(aggregate(delays.df$Flight.Status == "delayed", by = list(delays.df$DAY_WEEK), mean, rm.na =
T)[,2], xlab = "Day of Week", ylab = "Average Delay",
names.arg = c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"))
```

Code for generating heatmap for exploring flight delays. Percent of delayed flights (darker = higher %delays) by day of week, origin, and carrier (Figure 10.7)

```
####
library(reshape)
library(ggplot2)
# create matrix for plot
delays.df$isDelay <- 1 * (delays.df$Flight.Status == "delayed")

agg <- aggregate(delays.df$isDelay,
            by = list(delays.df$DAY_WEEK, delays.df$CARRIER, delays.df$ORIGIN),
            FUN = mean, na.rm = TRUE)
m <- melt(agg)
names(m)[1:3] <- c("DAY_WEEK", "CARRIER", "ORIGIN")

# plot with ggplot
# use facet_grid() with arguments scales = "free" and space = "free" to skip
# missing values.
ggplot(m, aes(y = CARRIER, x = DAY_WEEK, fill = value)) +  geom_tile() +
  facet_grid(ORIGIN ~ ., scales = "free", space = "free") +
  scale_fill_gradient(low="white", high="black")
```

Code for data preprocessing and running logistic regression. Estimated Logistic regression model for delayed flights (based on the training set) (Figure 10.8))

```
####
delays.df <- read.csv("FlightDelays.csv")

# transform variables and create bins
delays.df$DAY_WEEK <- factor(delays.df$DAY_WEEK, levels = c(1:7),
                    labels = c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"))
delays.df$CRS_DEP_TIME <- factor(round(delays.df$CRS_DEP_TIME/100))

# create reference categories
delays.df$ORIGIN <- relevel(delays.df$ORIGIN, ref = "IAD")
delays.df$DEST <- relevel(delays.df$DEST, ref = "LGA")
delays.df$CARRIER <- relevel(delays.df$CARRIER, ref = "US")
delays.df$DAY_WEEK <- relevel(delays.df$DAY_WEEK, ref = "Wed")
delays.df$isDelay <- 1 * (delays.df$Flight.Status == "delayed")

# create training and validation sets
selected.var <- c(10, 1, 8, 4, 2, 9, 14)
train.index <- sample(c(1:dim(delays.df)[1]), dim(delays.df)[1]*0.6)
train.df <- delays.df[train.index, selected.var]
valid.df <- delays.df[-train.index, selected.var]

# run logistic model, and show coefficients and odds
lm.fit <- glm(isDelay ~ ., data = train.df, family = "binomial")
```

```
data.frame(summary(lm.fit)$coefficients, odds = exp(coef(lm.fit)))

round(data.frame(summary(lm.fit)$coefficients, odds = exp(coef(lm.fit))), 5)
```

Code for evaluating performance of all-predictor model. Confusion matrix and lift chart for the flight delay validation data using all predictors (Figure 10.9))

```
####
library(gains)
pred <- predict(lm.fit, valid.df)
gain <- gains(valid.df$isDelay, pred, groups=100)

plot(c(0,gain$cume.pct.of.total*sum(valid.df$isDelay))~
     c(0,gain$cume.obs),
   xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.df$isDelay))~c(0, dim(valid.df)[1]), lty=2)
```

Code for logistic regression with fewer predictors. Logistic regression model with fewer predictors (Figure 10.11)

```
####
# fewer predictors
delays.df$Weekend <- delays.df$DAY_WEEK %in% c("Sun", "Sat")
delays.df$CARRIER_CO_MQ_DH_RU <- delays.df$CARRIER %in% c("CO", "MQ", "DH", "RU")
delays.df$MORNING <- delays.df$CRS_DEP_TIME %in% c(6, 7, 8, 9)
delays.df$NOON <- delays.df$CRS_DEP_TIME %in% c(10, 11, 12, 13)
delays.df$AFTER2P <- delays.df$CRS_DEP_TIME %in% c(14, 15, 16, 17, 18)
delays.df$EVENING <- delays.df$CRS_DEP_TIME %in% c(19, 20)

set.seed(1)  # Set the seed for the random number generator for reproducing the
# partition.
train.index <- sample(c(1:dim(delays.df)[1]), dim(delays.df)[1]*0.6)
valid.index <- setdiff(c(1:dim(delays.df)[1]), train.index)
train.df <- delays.df[train.index, ]
valid.df <- delays.df[valid.index, ]

lm.fit <- glm(isDelay ~ Weekend + Weather + CARRIER_CO_MQ_DH_RU + MORNING  +  NOON +
          AFTER2P + EVENING, data = train.df, family = "binomial")
summary(lm.fit)

# evaluate
pred <- predict(lm.fit, valid.df)
confusionMatrix(ifelse(pred > 0.5, 1, 0), valid.df$isDelay)
```

Output for multiple linear regression model of Personal Loan on three predictors (Figure 10.12)

```
####
reg <- lm(Personal.Loan ~ Income + Family + CD.Account, data = bank.df)
summary(reg)
```

Measures of explanatory power for Universal Bank training data with a 12 predictor model (Figure 10.15)

```
####
summary(logit.reg)
```

Confusion matrix and lift chart for Universal Bank training data with 12 predictors (Figure 10.15)

```
####
## note: run after Table 10.3, before adding more variables
confusionMatrix(ifelse(logit.reg$fitted > 0.5, 1, 0), train.df[,8])
```

Code for logistic regression with more than 2 classes. Ordinal and nominal multinomial regression in R (Appendix C)

```
####
# simulate simple data
Y = rep(c("a", "b", "c"), 100)
x = rep(c(1, 2, 3), 100) +  rnorm(300, 0, 1)

# ordinal logistic regression
library(MASS)
Y = factor(Y, ordered = T)
polr(Y ~ x)

# nominal logistic regression
library(nnet)
Y = factor(Y, ordered = F)
multinom(Y ~ x)
```