Ch5 Beginning Chapter Randall Plyler

Randall Plyler

1/24/2022

```
#### Table 5.1

# package forecast is required to evaluate performance
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
# load file
toyota.corolla.df <- read.csv("C:/Users/randa/Dropbox/Masters/Winter/TBANLT 560 Data Mining/Files/DMBA-

# randomly generate training and validation sets
training <- sample(toyota.corolla.df$Id, 600)
validation <- sample(setdiff(toyota.corolla.df$Id, training), 400)

#show(toyota.corolla.df)
# run linear regression model
reg <- lm(Price~., data=toyota.corolla.df[,-c(1,2,8,11)], subset=training,
          na.action=na.exclude)
#show(reg)
pred_t <- predict(reg, na.action=na.pass)
pred_v <- predict(reg, newdata=toyota.corolla.df[validation,-c(1,2,8,11)],
                  na.action=na.pass)
```

```
## Warning in predict.lm(reg, newdata = toyota.corolla.df[validation, -c(1, :
## prediction from a rank-deficient fit may be misleading
```

```
## evaluate performance
# training
accuracy(pred_t, toyota.corolla.df[training,]$Price)
```

```
##                        ME     RMSE      MAE        MPE     MAPE
## Test set -1.214923e-11 1006.367 759.3663 -0.8816135 7.648384
```

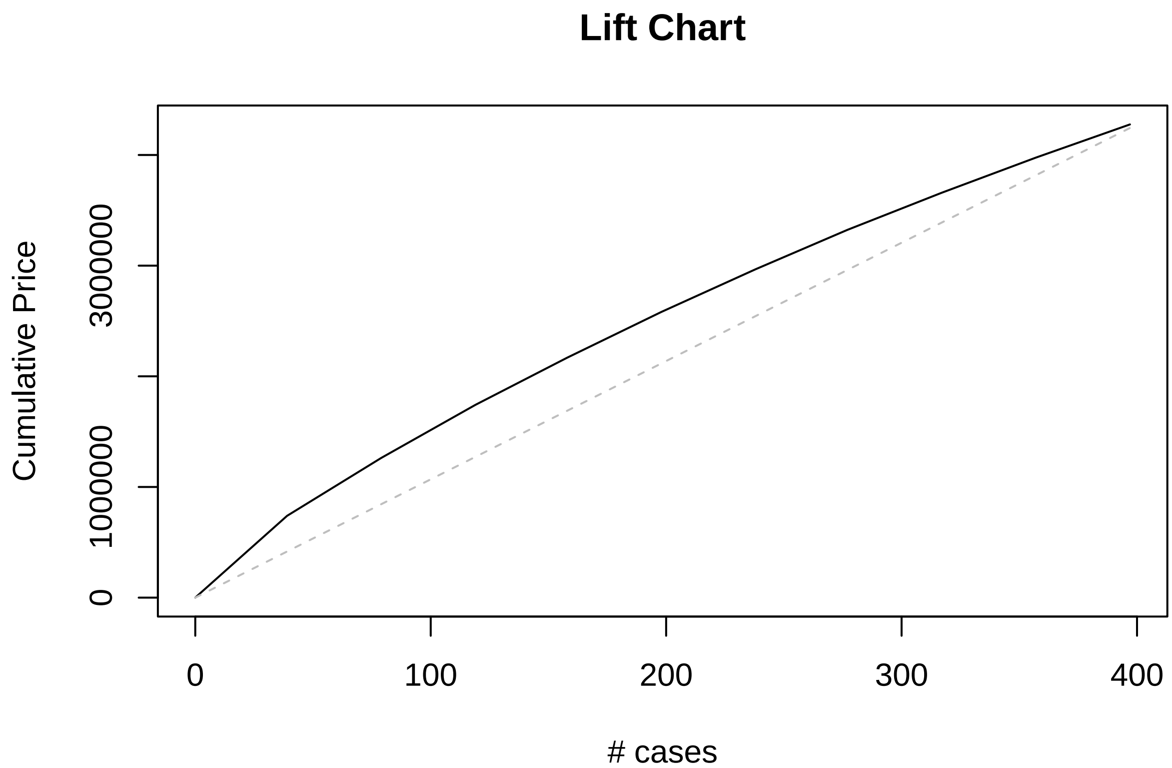
```
# validation
accuracy(pred_v, toyota.corolla.df[validation,]$Price)
```

```
##                ME     RMSE      MAE       MPE     MAPE
## Test set 49.5609 2710.891 959.9476 -1.271814 9.088691
```

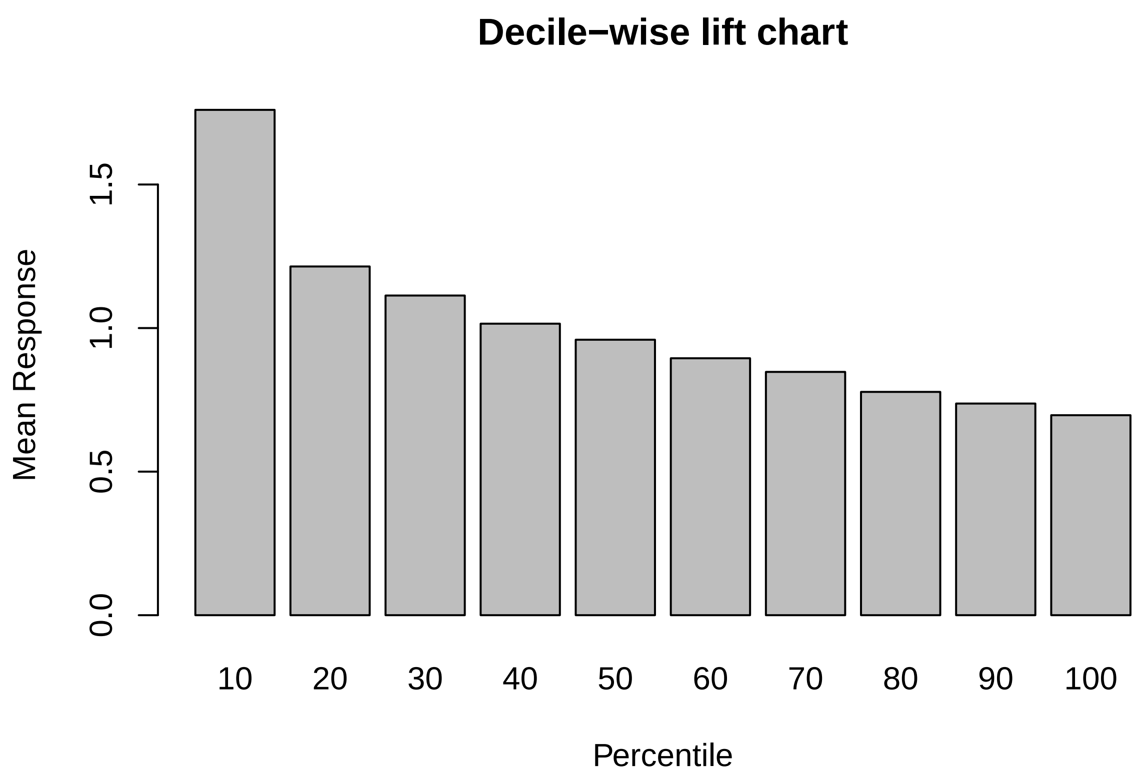
```r
#### Figure 5.2

# remove missing Price data
toyota.corolla.df <-
  toyota.corolla.df[!is.na(toyota.corolla.df[validation,]$Price),]

# generate random Training and Validation sets
training <- sample(toyota.corolla.df$Id, 600)
validation <- sample(toyota.corolla.df$Id, 400)

# regression model based on all numerical predictors
reg <- lm(Price~., data = toyota.corolla.df[,-c(1,2,8,11)], subset = training)

# predictions
pred_v <- predict(reg, newdata = toyota.corolla.df[validation,-c(1,2,8,11)])
```

```
## Warning in predict.lm(reg, newdata = toyota.corolla.df[validation, -c(1, :
## prediction from a rank-deficient fit may be misleading
```

```r
# load package gains, compute gains (we will use package caret for categorical y later)

library(gains)
gain <- gains(toyota.corolla.df[validation,]$Price[!is.na(pred_v)], pred_v[!is.na(pred_v)])

# cumulative lift chart
options(scipen=999) # avoid scientific notation
# we will compute the gain relative to price
price <- toyota.corolla.df[validation,]$Price[!is.na(toyota.corolla.df[validation,]$Price)]
plot(c(0,gain$cume.pct.of.total*sum(price))~c(0,gain$cume.obs),
     xlab="# cases", ylab="Cumulative Price", main="Lift Chart", type="l")

# baseline
lines(c(0,sum(price))~c(0,dim(toyota.corolla.df[validation,])[1]), col="gray", lty=2)
```




```r
# Decile-wise lift chart
barplot(gain$mean.resp/mean(price), names.arg = gain$depth,
        xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
```




```r
#### Table 5.5

library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(e1071)

owner.df <- read.csv("C:/Users/randa/Dropbox/Masters/Winter/TBANLT 560 Data Mining/Files/DMBA-R-datasets
head(owner.df)
```

```
##   Class Probability
## 1 owner      0.9959
## 2 owner      0.9875
## 3 owner      0.9844
## 4 owner      0.9804
## 5 owner      0.9481
## 6 owner      0.8892
```

```r
confusionMatrix(as.factor(ifelse(owner.df$Probability>0.5, 'owner', 'nonowner')),
                as.factor(owner.df$Class))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction nonowner owner
##   nonowner       10     1
##   owner           2    11
##
##                Accuracy : 0.875
##                  95% CI : (0.6764, 0.9734)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0001386
##
##                   Kappa : 0.75
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.8333
##             Specificity : 0.9167
##          Pos Pred Value : 0.9091
##          Neg Pred Value : 0.8462
##              Prevalence : 0.5000
##          Detection Rate : 0.4167
##    Detection Prevalence : 0.4583
##       Balanced Accuracy : 0.8750
##
##        'Positive' Class : nonowner
##
```

```
confusionMatrix(as.factor(ifelse(owner.df$Probability>0.25, 'owner', 'nonowner')),
                as.factor(owner.df$Class))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction nonowner owner
##   nonowner        8     1
##   owner           4    11
##
##                Accuracy : 0.7917
##                  95% CI : (0.5785, 0.9287)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.003305
##
##                   Kappa : 0.5833
##
##  Mcnemar's Test P-Value : 0.371093
##
##             Sensitivity : 0.6667
##             Specificity : 0.9167
##          Pos Pred Value : 0.8889
##          Neg Pred Value : 0.7333
##              Prevalence : 0.5000
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.3750
```

```
##       Balanced Accuracy : 0.7917
##
##        'Positive' Class : nonowner
##
```

```r
confusionMatrix(as.factor(ifelse(owner.df$Probability>0.75, 'owner', 'nonowner')),
                as.factor(owner.df$Class))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction nonowner owner
##   nonowner       11     5
##   owner           1     7
##
##                Accuracy : 0.75
##                  95% CI : (0.5329, 0.9023)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.01133
##
##                   Kappa : 0.5
##
##  Mcnemar's Test P-Value : 0.22067
##
##             Sensitivity : 0.9167
##             Specificity : 0.5833
##          Pos Pred Value : 0.6875
##          Neg Pred Value : 0.8750
##              Prevalence : 0.5000
##          Detection Rate : 0.4583
##    Detection Prevalence : 0.6667
##       Balanced Accuracy : 0.7500
##
##        'Positive' Class : nonowner
##
```

```r
as.factor(ifelse(owner.df$Probability>0.5, 'owner', 'nonowner'))
```

```
##  [1] owner    owner    owner    owner    owner    owner    owner    owner
##  [9] owner    owner    owner    owner    owner    nonowner nonowner nonowner
## [17] nonowner nonowner nonowner nonowner nonowner nonowner nonowner nonowner
## Levels: nonowner owner
```

```r
#### Figure 5.4

# replace data.frame with your own
df <- read.csv("C:/Users/randa/Dropbox/Masters/Winter/TBANLT 560 Data Mining/Files/DMBA-R-datasets/DMBA

head(df)
```

```
##    prob actual  X X.1 X.2
## 1 0.995      1 NA  NA  NA
```

```
## 2 0.998      1 NA  NA  NA
## 3 0.985      1 NA  NA  NA
## 4 0.980      1 NA  NA  NA
## 5 0.948      1 NA  NA  NA
## 6 0.889      1 NA  NA  NA
```

```r
# create empty accuracy table
accT = c()
cut=1
cm <- confusionMatrix(as.factor(1 * (df$prob > cut)), as.factor(df$actual))
```

```
## Warning in confusionMatrix.default(as.factor(1 * (df$prob > cut)),
## as.factor(df$actual)): Levels are not in the same order for reference and data.
## Refactoring data to match.
```

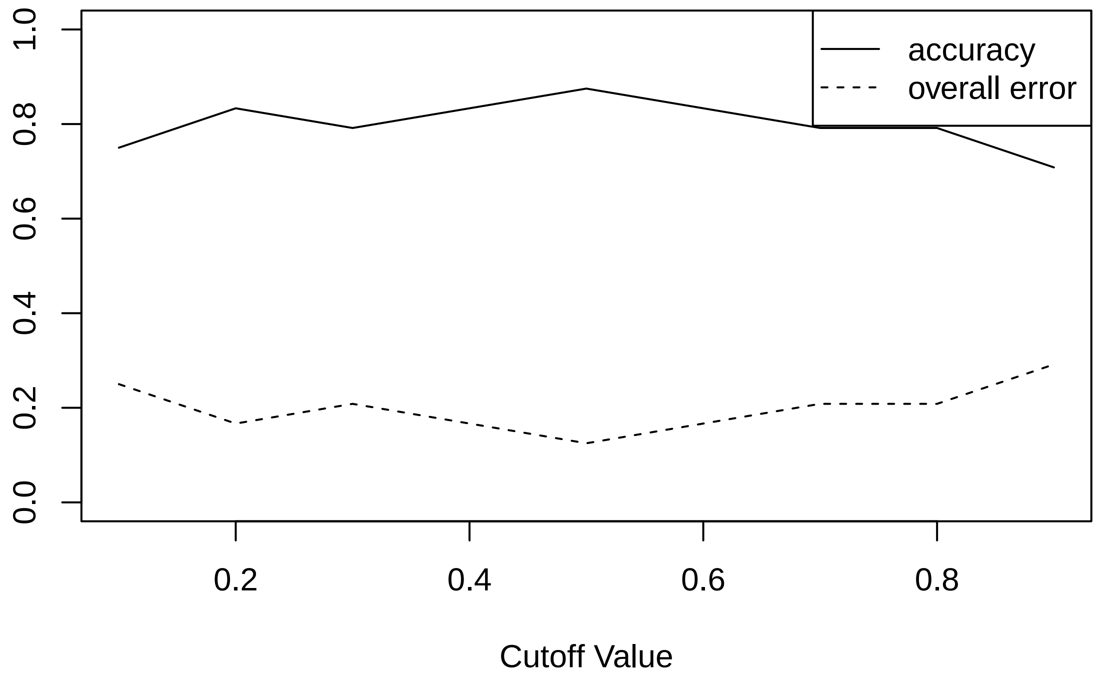
```r
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 12 12
##          1  0  0
##
##                Accuracy : 0.5
##                  95% CI : (0.2912, 0.7088)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.580590
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.001496
##
##             Sensitivity : 1.0
##             Specificity : 0.0
##          Pos Pred Value : 0.5
##          Neg Pred Value : NaN
##              Prevalence : 0.5
##          Detection Rate : 0.5
##    Detection Prevalence : 1.0
##       Balanced Accuracy : 0.5
##
##        'Positive' Class : 0
##
```

```r
# compute accuracy per cutoff
for (cut in seq(0.1,.9,0.1)){
  cm <- confusionMatrix(as.factor(1 * (df$prob > cut)), as.factor(df$actual))
  accT = c(accT, cm$overall[1])
}
length(accT)
```

```
## [1] 9
```

```r
# plot accuracy
plot(accT ~ seq(0.1,.9,0.1), xlab = "Cutoff Value", ylab = "", type = "l", ylim = c(0, 1))
lines(1-accT ~ seq(0.1,.9,0.1), type = "l", lty = 2)
legend("topright",  c("accuracy", "overall error"), lty = c(1, 2), merge = TRUE)
```



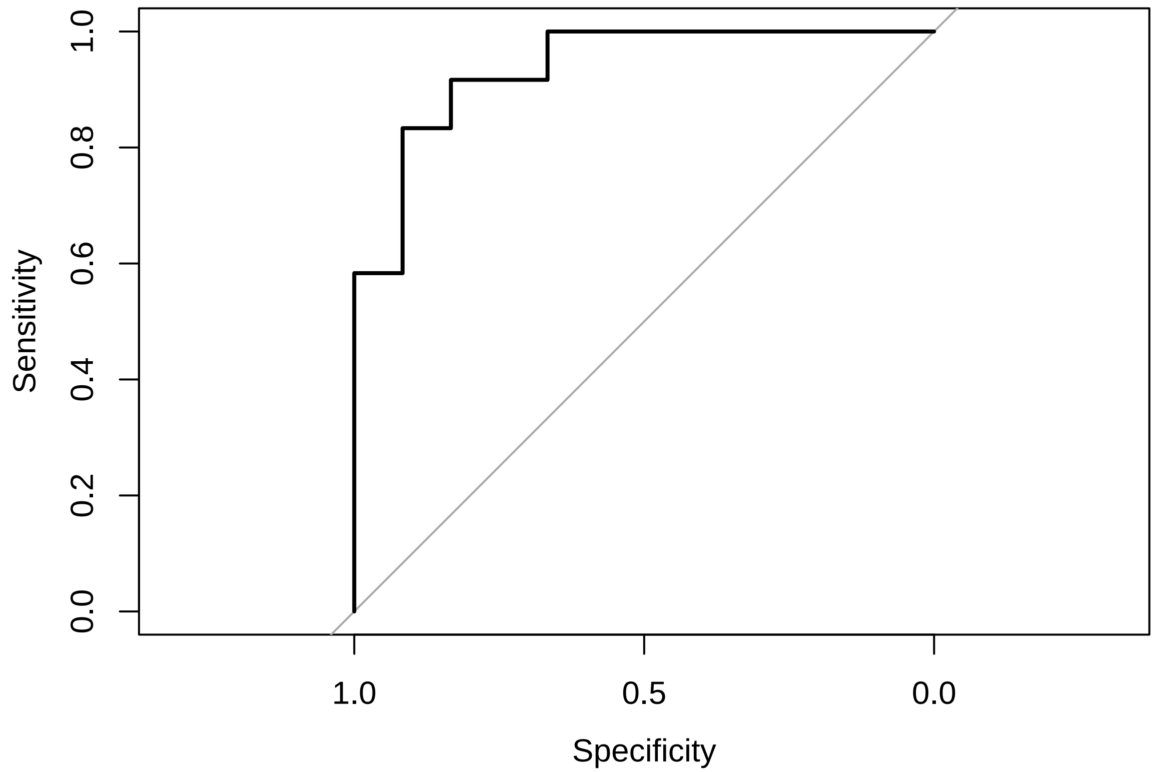

```r
#### Figure 5.5

library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
r <- roc(df$actual, df$prob)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```r
plot.roc(r)
```




```r
# compute auc
auc(r)
```

```
## Area under the curve: 0.9375
```

```r
#### Figure 5.6

# first option with 'caret' library:
library(caret)
lift.example <- lift(relevel(as.factor(actual), ref="1") ~ prob, data = df)

df[1:10,]
```

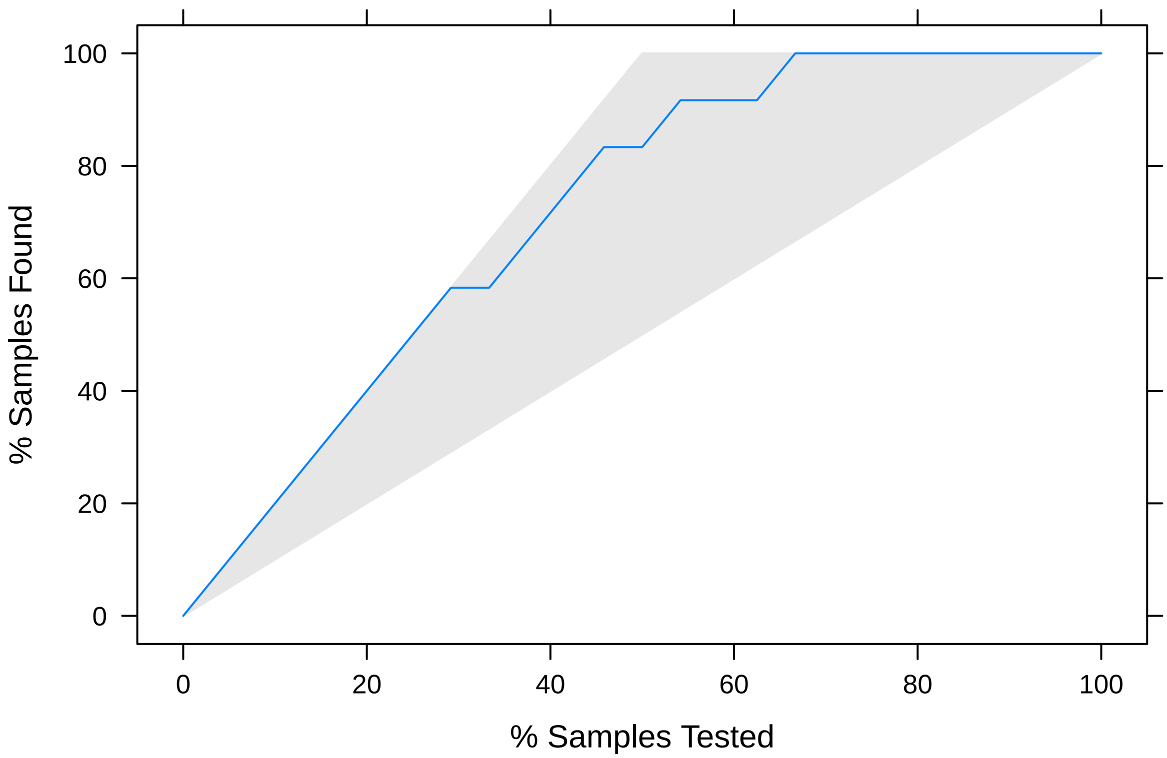
```
##     prob actual  X X.1 X.2
## 1  0.995      1 NA  NA  NA
## 2  0.998      1 NA  NA  NA
## 3  0.985      1 NA  NA  NA
## 4  0.980      1 NA  NA  NA
## 5  0.948      1 NA  NA  NA
## 6  0.889      1 NA  NA  NA
## 7  0.847      1 NA  NA  NA
## 8  0.762      0 NA  NA  NA
```

```
## 9  0.706      1 NA  NA  NA
## 10 0.680      1 NA  NA  NA
```

```
tail(df)
```

```
##     prob actual  X X.1 X.2
## 19 0.047      0 NA  NA  NA
## 20 0.038      0 NA  NA  NA
## 21 0.024      0 NA  NA  NA
## 22 0.021      0 NA  NA  NA
## 23 0.016      0 NA  NA  NA
## 24 0.003      0 NA  NA  NA
```

```
xyplot(lift.example, plot = "gain")
```



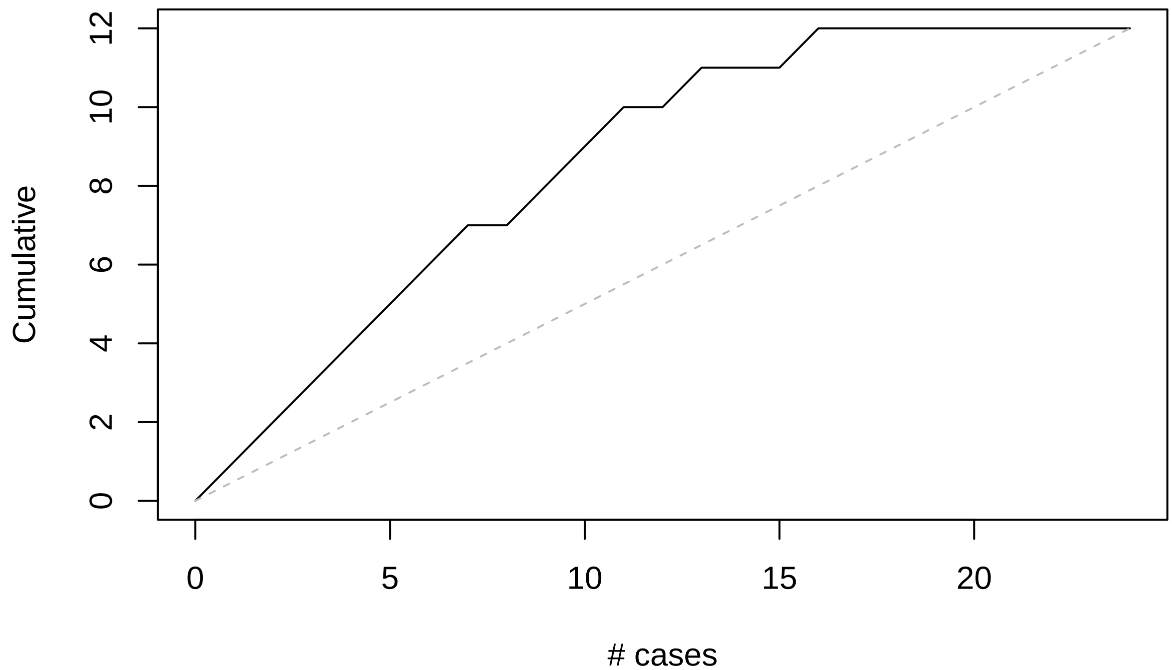

```
lift.example$data$EventPct
```

```
##  [1]   0.00000 100.00000 100.00000 100.00000 100.00000 100.00000 100.00000
##  [8] 100.00000  87.50000  88.88889  90.00000  90.90909  83.33333  84.61538
## [15]  78.57143  73.33333  75.00000  70.58824  66.66667  63.15789  60.00000
## [22]  57.14286  54.54545  52.17391  50.00000  50.00000
```

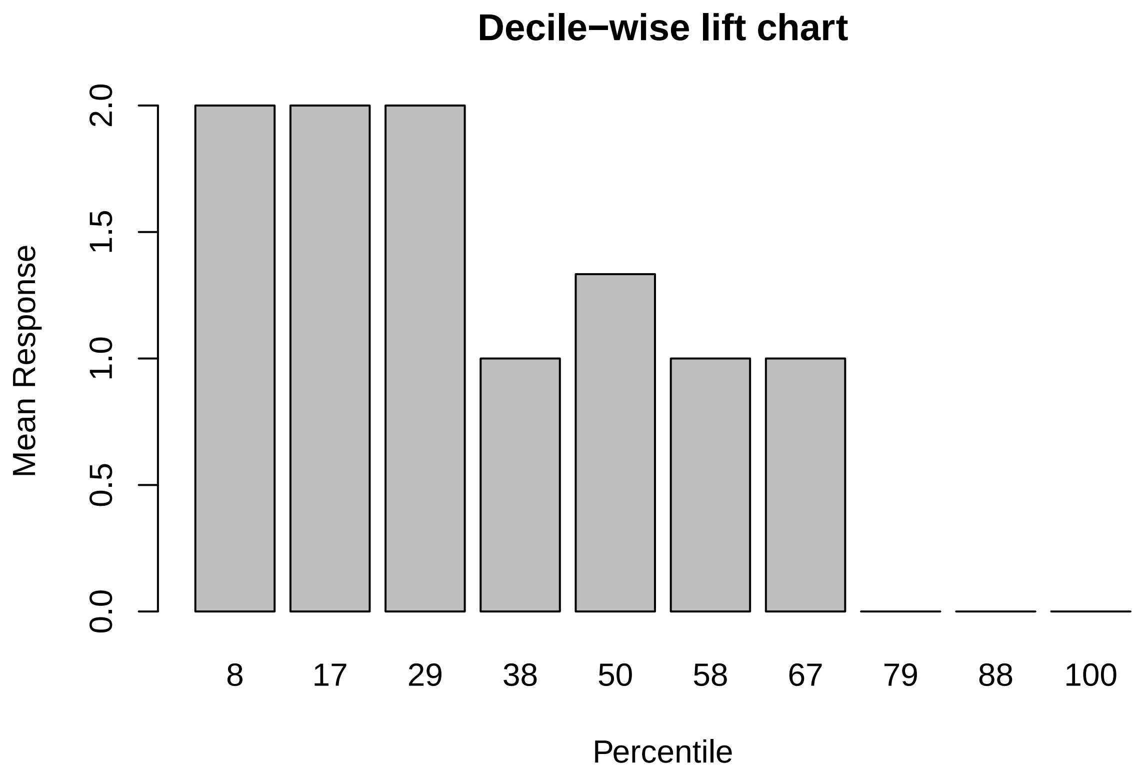
```r
str(lift.example)
```

```
## List of 5
##  $ data     :'data.frame':   26 obs. of  10 variables:
##   ..$ liftModelVar: chr [1:26] "prob" "prob" "prob" "prob" ...
##   ..$ cuts        : num [1:26] 1 0.998 0.995 0.985 0.98 0.948 0.889 0.847 0.762 0.706 ...
##   ..$ events      : int [1:26] 0 1 2 3 4 5 6 7 7 8 ...
##   ..$ n           : int [1:26] 0 1 2 3 4 5 6 7 8 9 ...
##   ..$ Sn          : num [1:26] 0 0.0833 0.1667 0.25 0.3333 ...
##   ..$ Sp          : num [1:26] 1 1 1 1 1 ...
##   ..$ EventPct    : num [1:26] 0 100 100 100 100 ...
##   ..$ CumEventPct : num [1:26] 0 8.33 16.67 25 33.33 ...
##   ..$ lift        : num [1:26] NaN 2 2 2 2 ...
##   ..$ CumTestedPct: num [1:26] 0 4.17 8.33 12.5 16.67 ...
##  $ class    : chr "1"
##  $ probNames: chr "prob"
##  $ pct      : num 50
##  $ call     : language lift.formula(x = relevel(as.factor(actual), ref = "1") ~ prob, data = df)
##  - attr(*, "class")= chr "lift"
```

```r
# Second option with 'gains' library:
library(gains)
df <- read.csv("C:/Users/randa/Dropbox/Masters/Winter/TBANLT 560 Data Mining/Files/DMBA-R-datasets/DMBA
gain <- gains(df$actual, df$prob, groups=dim(df)[1])
plot(c(0, gain$cume.pct.of.total*sum(df$actual)) ~ c(0, gain$cume.obs),
     xlab = "# cases", ylab = "Cumulative", type="l")
lines(c(0,sum(df$actual))~c(0,dim(df)[1]), col="gray", lty=2)
```




```
#### Figure 5.7

# use gains() to compute deciles.
# when using the caret package, deciles must be computed manually.

gain <- gains(df$actual, df$prob)
barplot(gain$mean.resp / mean(df$actual), names.arg = gain$depth, xlab = "Percentile",
        ylab = "Mean Response", main = "Decile-wise lift chart")
```


Decile−wise lift chart
Mean Response
2.0
1.5
1.0
0.5
0.0
8
17
29
38
50
58
67
79
88
100
Percentile