BUAN6356_Homework3_NarangM

Narang,Mandeep

11/16/2018

installing required packages

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(gains)
library(gains)
library(MASS)
library(tidyverse)
```

```
## ── Attaching packages ────────────────────────────────────────
───── tidyverse 1.2.1 ──
```

```
## ✔ tibble  1.4.2     ✔ purrr   0.2.5
## ✔ tidyr   0.8.1     ✔ dplyr   0.7.8
## ✔ readr   1.1.1     ✔ stringr 1.3.1
## ✔ tibble  1.4.2     ✔ forcats 0.3.0
```

```
## ── Conflicts ─────────────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
## ✖ dplyr::select() masks MASS::select()
```

```
library(dplyr)
```

reading the Dataset online as a Table without Headers

```
Data.spam <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/sp
ambase/spambase.data",
                      header = FALSE,
                      sep = ",")
```

Having a look on the Structure of the Dataset

```
head(Data.spam)
```

```
##     V1   V2   V3 V4   V5   V6   V7   V8   V9  V10  V11  V12  V13  V14  V15
## 1 0.00 0.64 0.64  0 0.32 0.00 0.00 0.00 0.00 0.00 0.00 0.64 0.00 0.00 0.00
## 2 0.21 0.28 0.50  0 0.14 0.28 0.21 0.07 0.00 0.94 0.21 0.79 0.65 0.21 0.14
## 3 0.06 0.00 0.71  0 1.23 0.19 0.19 0.12 0.64 0.25 0.38 0.45 0.12 0.00 1.75
## 4 0.00 0.00 0.00  0 0.63 0.00 0.31 0.63 0.31 0.63 0.31 0.31 0.31 0.00 0.00
## 5 0.00 0.00 0.00  0 0.63 0.00 0.31 0.63 0.31 0.63 0.31 0.31 0.31 0.00 0.00
## 6 0.00 0.00 0.00  0 1.85 0.00 0.00 1.85 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##    V16  V17  V18  V19  V20  V21 V22  V23  V24 V25 V26 V27 V28 V29 V30 V31
## 1 0.32 0.00 1.29 1.93 0.00 0.96   0 0.00 0.00   0   0   0   0   0   0   0
## 2 0.14 0.07 0.28 3.47 0.00 1.59   0 0.43 0.43   0   0   0   0   0   0   0
## 3 0.06 0.06 1.03 1.36 0.32 0.51   0 1.16 0.06   0   0   0   0   0   0   0
## 4 0.31 0.00 0.00 3.18 0.00 0.31   0 0.00 0.00   0   0   0   0   0   0   0
## 5 0.31 0.00 0.00 3.18 0.00 0.31   0 0.00 0.00   0   0   0   0   0   0   0
## 6 0.00 0.00 0.00 0.00 0.00 0.00   0 0.00 0.00   0   0   0   0   0   0   0
##   V32 V33 V34 V35 V36  V37 V38 V39  V40 V41 V42  V43 V44  V45  V46 V47 V48
## 1   0   0   0   0   0 0.00   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0
## 2   0   0   0   0   0 0.07   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0
## 3   0   0   0   0   0 0.00   0   0 0.06   0   0 0.12   0 0.06 0.06   0   0
## 4   0   0   0   0   0 0.00   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0
## 5   0   0   0   0   0 0.00   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0
## 6   0   0   0   0   0 0.00   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0
##    V49   V50 V51   V52   V53   V54   V55 V56  V57 V58
## 1 0.00 0.000   0 0.778 0.000 0.000 3.756  61  278   1
## 2 0.00 0.132   0 0.372 0.180 0.048 5.114 101 1028   1
## 3 0.01 0.143   0 0.276 0.184 0.010 9.821 485 2259   1
## 4 0.00 0.137   0 0.137 0.000 0.000 3.537  40  191   1
## 5 0.00 0.135   0 0.135 0.000 0.000 3.537  40  191   1
## 6 0.00 0.223   0 0.000 0.000 0.000 3.000  15   54   1
```

```
tail(Data.spam)
```

```
##        V1 V2   V3 V4   V5   V6 V7 V8 V9 V10 V11  V12  V13 V14 V15 V16 V17
## 4596 0.00  0 1.19  0 0.00 0.00  0  0  0   0   0 0.00 0.00   0   0   0   0
## 4597 0.31  0 0.62  0 0.00 0.31  0  0  0   0   0 1.88 0.00   0   0   0   0
## 4598 0.00  0 0.00  0 0.00 0.00  0  0  0   0   0 0.00 0.00   0   0   0   0
## 4599 0.30  0 0.30  0 0.00 0.00  0  0  0   0   0 1.80 0.30   0   0   0   0
## 4600 0.96  0 0.00  0 0.32 0.00  0  0  0   0   0 0.32 0.00   0   0   0   0
## 4601 0.00  0 0.65  0 0.00 0.00  0  0  0   0   0 0.00 0.65   0   0   0   0
##       V18  V19 V20  V21 V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33
## 4596 0.59 3.57   0 1.19   0   0   0   0   0   0   0   0   0   0   0   0
## 4597 0.00 0.62   0 0.00   0   0   0   0   0   0   0   0   0   0   0   0
## 4598 0.00 6.00   0 2.00   0   0   0   0   0   0   0   0   0   0   0   0
## 4599 0.90 1.50   0 0.30   0   0   0   0   0   0   0   0   0   0   0   0
## 4600 0.00 1.93   0 0.32   0   0   0   0   0   0   0   0   0   0   0   0
## 4601 0.00 4.60   0 0.65   0   0   0   0   0   0   0   0   0   0   0   0
##      V34 V35 V36 V37 V38 V39 V40 V41 V42 V43  V44  V45  V46 V47 V48   V49
## 4596   0   0   0   0   0   0   0   0   0   0 0.00 0.00 0.59   0   0 0.000
## 4597   0   0   0   0   0   0   0   0   0   0 0.31 0.31 0.31   0   0 0.000
## 4598   0   0   0   0   0   0   0   0   0   0 0.00 0.00 2.00   0   0 0.000
## 4599   0   0   0   0   0   0   0   0   0   0 0.00 0.00 1.20   0   0 0.102
## 4600   0   0   0   0   0   0   0   0   0   0 0.32 0.00 0.32   0   0 0.000
## 4601   0   0   0   0   0   0   0   0   0   0 0.00 1.97 0.65   0   0 0.000
##        V50 V51   V52 V53 V54   V55 V56 V57 V58
## 4596 0.000   0 0.000   0   0 1.000   1  24   0
## 4597 0.232   0 0.000   0   0 1.142   3  88   0
## 4598 0.000   0 0.353   0   0 1.555   4  14   0
## 4599 0.718   0 0.000   0   0 1.404   6 118   0
## 4600 0.057   0 0.000   0   0 1.147   5  78   0
## 4601 0.000   0 0.125   0   0 1.250   5  40   0
```

```
glimpse(Data.spam)
```

```
## Observations: 4,601
## Variables: 58
## $ V1  <dbl> 0.00, 0.21, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.15, 0.06...
## $ V2  <dbl> 0.64, 0.28, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.12...
## $ V3  <dbl> 0.64, 0.50, 0.71, 0.00, 0.00, 0.00, 0.00, 0.00, 0.46, 0.77...
## $ V4  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V5  <dbl> 0.32, 0.14, 1.23, 0.63, 0.63, 1.85, 1.92, 1.88, 0.61, 0.19...
## $ V6  <dbl> 0.00, 0.28, 0.19, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.32...
## $ V7  <dbl> 0.00, 0.21, 0.19, 0.31, 0.31, 0.00, 0.00, 0.00, 0.30, 0.38...
## $ V8  <dbl> 0.00, 0.07, 0.12, 0.63, 0.63, 1.85, 0.00, 1.88, 0.00, 0.00...
## $ V9  <dbl> 0.00, 0.00, 0.64, 0.31, 0.31, 0.00, 0.00, 0.00, 0.92, 0.06...
## $ V10 <dbl> 0.00, 0.94, 0.25, 0.63, 0.63, 0.00, 0.64, 0.00, 0.76, 0.00...
## $ V11 <dbl> 0.00, 0.21, 0.38, 0.31, 0.31, 0.00, 0.96, 0.00, 0.76, 0.00...
## $ V12 <dbl> 0.64, 0.79, 0.45, 0.31, 0.31, 0.00, 1.28, 0.00, 0.92, 0.64...
## $ V13 <dbl> 0.00, 0.65, 0.12, 0.31, 0.31, 0.00, 0.00, 0.00, 0.00, 0.25...
## $ V14 <dbl> 0.00, 0.21, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V15 <dbl> 0.00, 0.14, 1.75, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.12...
## $ V16 <dbl> 0.32, 0.14, 0.06, 0.31, 0.31, 0.00, 0.96, 0.00, 0.00, 0.00...
## $ V17 <dbl> 0.00, 0.07, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V18 <dbl> 1.29, 0.28, 1.03, 0.00, 0.00, 0.00, 0.32, 0.00, 0.15, 0.12...
## $ V19 <dbl> 1.93, 3.47, 1.36, 3.18, 3.18, 0.00, 3.85, 0.00, 1.23, 1.67...
## $ V20 <dbl> 0.00, 0.00, 0.32, 0.00, 0.00, 0.00, 0.00, 0.00, 3.53, 0.06...
## $ V21 <dbl> 0.96, 1.59, 0.51, 0.31, 0.31, 0.00, 0.64, 0.00, 2.00, 0.71...
## $ V22 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V23 <dbl> 0.00, 0.43, 1.16, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.19...
## $ V24 <dbl> 0.00, 0.43, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.15, 0.00...
## $ V25 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V26 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V27 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V28 <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V29 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V30 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V31 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V32 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V33 <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.15, 0.00...
## $ V34 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V35 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V36 <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V37 <dbl> 0.00, 0.07, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V38 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V39 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V40 <dbl> 0.00, 0.00, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V41 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V42 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V43 <dbl> 0.00, 0.00, 0.12, 0.00, 0.00, 0.00, 0.00, 0.00, 0.30, 0.00...
## $ V44 <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.06...
## $ V45 <dbl> 0.00, 0.00, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V46 <dbl> 0.00, 0.00, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ V47 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V48 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V49 <dbl> 0.000, 0.000, 0.010, 0.000, 0.000, 0.000, 0.000, 0.000, 0....
## $ V50 <dbl> 0.000, 0.132, 0.143, 0.137, 0.135, 0.223, 0.054, 0.206, 0....
## $ V51 <dbl> 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0....
## $ V52 <dbl> 0.778, 0.372, 0.276, 0.137, 0.135, 0.000, 0.164, 0.000, 0....
## $ V53 <dbl> 0.000, 0.180, 0.184, 0.000, 0.000, 0.000, 0.054, 0.000, 0....
## $ V54 <dbl> 0.000, 0.048, 0.010, 0.000, 0.000, 0.000, 0.000, 0.000, 0....
## $ V55 <dbl> 3.756, 5.114, 9.821, 3.537, 3.537, 3.000, 1.671, 2.450, 9....
## $ V56 <int> 61, 101, 485, 40, 40, 15, 4, 11, 445, 43, 6, 11, 61, 7, 24...
## $ V57 <int> 278, 1028, 2259, 191, 191, 54, 112, 49, 1257, 749, 21, 184...
## $ V58 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
```

spam (1) or notspam (0)

Changing the V58 to Factors(we will use them as categories)

```
Data.spam$V58 <- factor(Data.spam$V58)
```

Calculating avgs for every variable by grouping them on V58

```
Data.spam.avg<-  Data.spam %>%group_by(V58) %>% summarise_all(funs(mean))
```

Gathering them and bringing columns to rows

finding the top 10 variables

```
  Data.spam.avgdiff<- Data.spam.avg %>% gather("Variable","Value",-V58)
  Data.spam.avgdiff<- Data.spam.avgdiff%>%  spread(key=V58, value=Value)
  Data.spam.avgdiff<- data.frame(Data.spam.avgdiff,abs(Data.spam.avgdiff[,3] - Data.s
pam.avgdiff[,2]))

  Data.spam.avgdiff.sorted <- arrange(Data.spam.avgdiff,desc(Data.spam.avgdiff$X1.1))
  topten<- Data.spam.avgdiff.sorted[1:10,1]
  topten<- c(topten,"V58")
```

Dataset after top ten selections

```
Data.spam.final <- Data.spam[,topten]
levels(Data.spam.final$V58) <- c("non spam","spam")
```

Now splitting the data into training (80%) and validation set (20%)

```
set.seed(123)
training.index <- createDataPartition(Data.spam.final$V58, p = 0.8, list = FALSE)
train.data <- Data.spam.final[training.index, ]
test.data <- Data.spam.final[-training.index, ]
```

Normalizing the data and estimating preprocessing parameters

```
normalized  <- preProcess(train.data, method = c("center", "scale"))
```

Now we will transform the data using the estimated parameters

```
train.norm <- predict(normalized, train.data)
test.norm <- predict(normalized, test.data)
```

Now running model on normalised data

```
lda_spam<- lda(V58 ~ ., data = train.norm)
lda_spam
```

```
## Call:
## lda(V58 ~ ., data = train.norm)
##
## Prior probabilities of groups:
##  non spam      spam
## 0.6059207 0.3940793
##
## Group means:
##                 V57        V56         V55        V27        V19
## non spam -0.1963386 -0.1621900 -0.09036762  0.1494132 -0.2085072
## spam      0.3018824  0.2493769  0.13894567 -0.2297318  0.3205924
##                 V21        V25        V16        V26        V52
## non spam -0.3112670  0.2041624 -0.2068699  0.1913071 -0.2322082
## spam      0.4785918 -0.3139120  0.3180749 -0.2941463  0.3570342
##
## Coefficients of linear discriminants:
##             LD1
## V57  0.37434917
## V56  0.09609042
## V55  0.06498994
## V27 -0.21887517
## V19  0.18779368
## V21  0.56321858
## V25 -0.22308671
## V16  0.38623072
## V26 -0.17508620
## V52  0.38510008
```

Predict propensities

```
prediction <- predict(lda_spam,test.norm[, -11], type = "response")
```

checking model accuracy

prediction v actual confusion matrix

```
table(prediction$class, test.norm$V58)
```

```
##
##             non spam spam
##   non spam      512  116
##   spam           45  246
```

```
mean(prediction$class == test.norm$V58)
```

```
## [1] 0.8248096
```

```
sum(prediction$posterior[, 1] >=.5) # cutoff lelel 0.5
```

```
## [1] 628
```

cumulative lift chart

```
gain <- gains(as.numeric(test.norm$V58), prediction$x[,1], groups = 10)

gain
```

```
## Depth                              Cume   Cume Pct                     Mean
##  of           Cume      Mean       Mean   of Total    Lift    Cume     Model
## File     N      N       Resp       Resp       Resp   Index    Lift     Score
## -------------------------------------------------------------------------
##   10    91     91       1.88       1.88      13.3%     135     135      2.35
##   20    92    183       1.86       1.87      26.7%     133     134      1.20
##   30    92    275       1.82       1.85      39.7%     130     133      0.74
##   40    92    367       1.63       1.80      51.4%     117     129      0.31
##   50    92    459       1.32       1.70      60.9%      94     122     -0.03
##   60    92    551       1.20       1.62      69.5%      86     116     -0.35
##   70    92    643       1.15       1.55      77.8%      83     111     -0.58
##   80    92    735       1.10       1.49      85.6%      79     107     -0.77
##   90    92    827       1.00       1.44      92.8%      72     103     -0.89
##  100    92    919       1.00       1.39     100.0%      72     100     -1.88
```

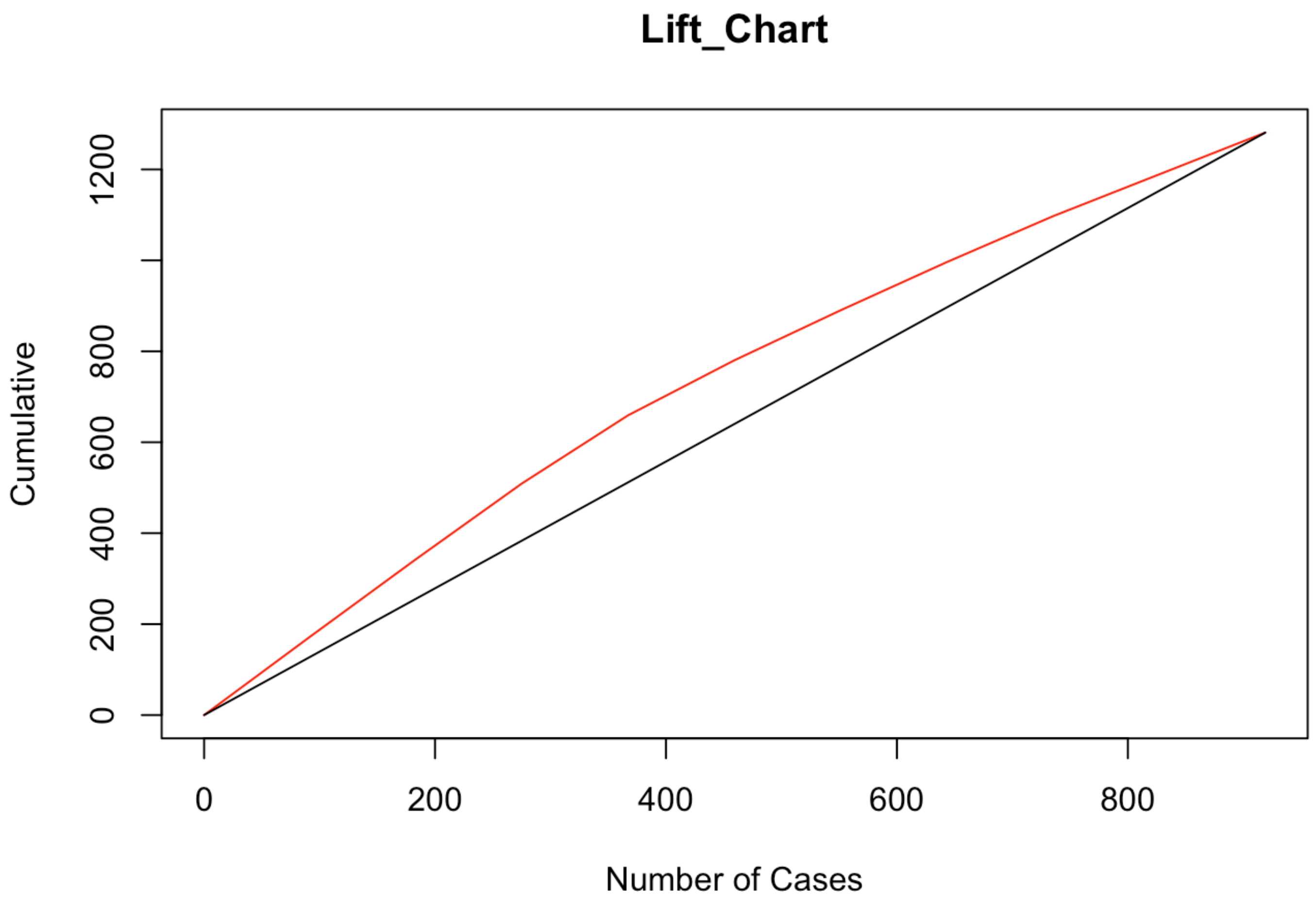
```
str(prediction$posterior)
```

```
##  num [1:919, 1:2] 0.241 0.211 0.866 0.562 0.583 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:919] "2" "3" "8" "10" ...
##   ..$ : chr [1:2] "non spam" "spam"
```

```
options(scipen=999)
```

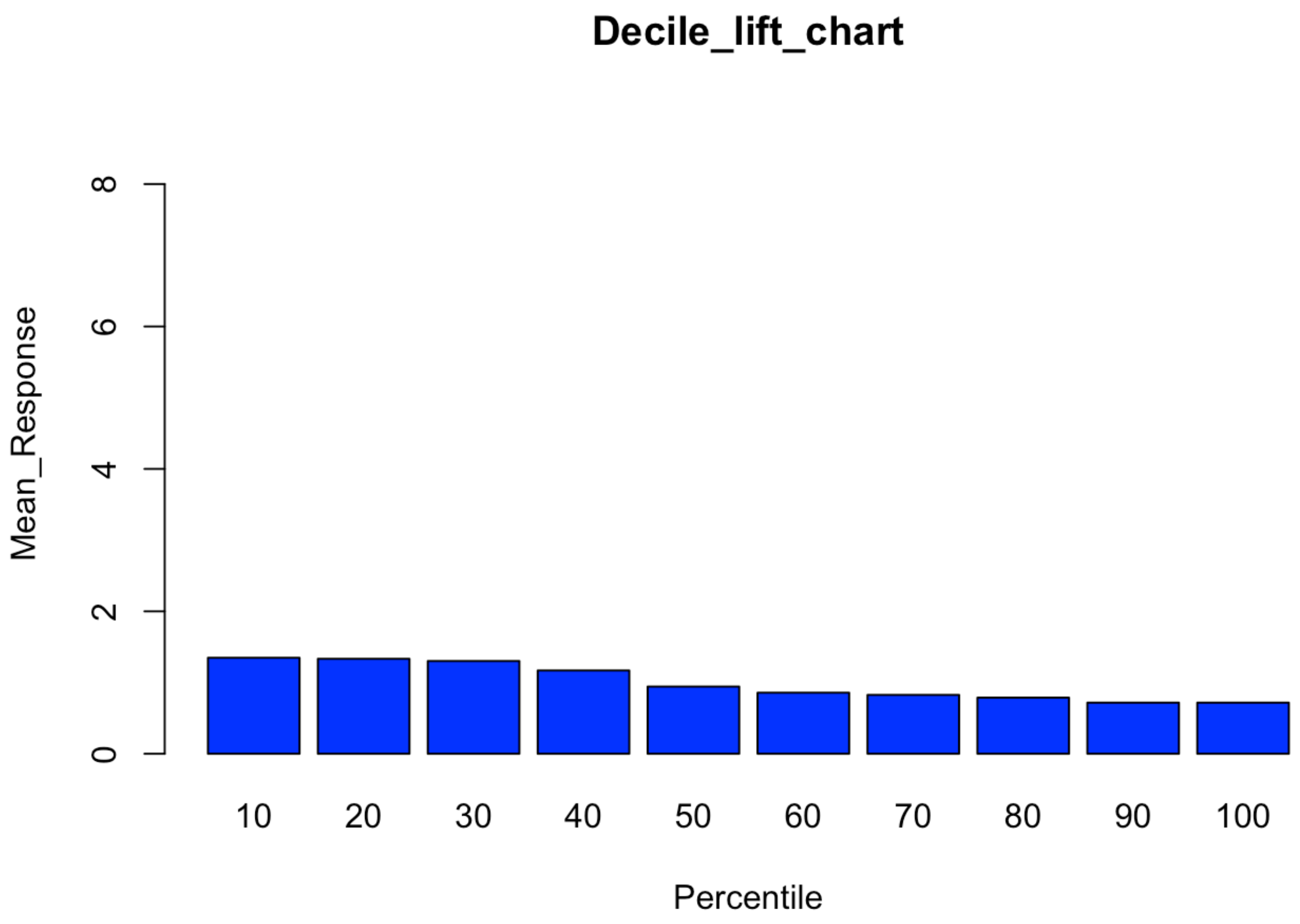
Compute gains relative to price baseline

```
spam<- as.numeric(test.norm$V58)
plot(c(0,gain$cume.pct.of.total*sum(spam))~c(0,gain$cume.obs),
     xlab="Number of Cases", ylab="Cumulative", main="Lift_Chart",
     col = "red", type="l")
lines(c(0,sum(spam))~c(0, dim(test.data)[1]), lty = 7)
```



Plot decile-wise chart

```
barheight <- gain$mean.resp/mean(spam)
midpoints <- barplot(barheight, names.arg = gain$depth,  ylim = c(0,9), col = "blue1"
,
                     xlab = "Percentile", ylab = "Mean_Response",
                     main = "Decile_lift_chart")
```


Decile_lift_chart
Mean_Response
Percentile