Chapter 8

Wenduo Wang

July 31, 2016

Problem 8

8. In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative variable.

```
## [1] 0
```

```
##          used (Mb) gc trigger (Mb) max used (Mb)
## Ncells 417401 22.3     750400 40.1   592000 31.7
## Vcells 657299  5.1    1308461 10.0   934317  7.2
```

First load necessary libraries

```
library(tree)
library(ISLR)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

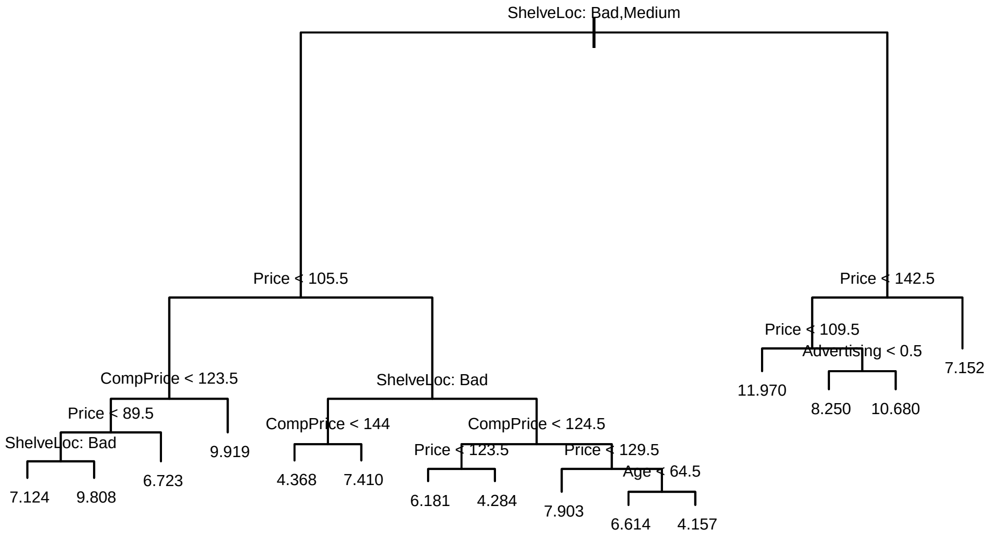
(a) Split the data set into a training set and a test set.

```
set.seed(2)
training_index <- sample(nrow(Carseats), 0.8*nrow(Carseats))
training_set <- Carseats[training_index,]
test_set <- Carseats[-training_index,]
```

(b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

The regression tree is summarised below, from the plot it is clear that `ShelveLoc` is an important predictor of `Sales` since it sits on the top of the tree. `Price` is also critical because it is included in more than one level.

```
tree_fit <- tree(Sales~., data=training_set)
summary(tree_fit)
plot(tree_fit)
text(tree_fit, pretty=0, cex=0.5)
```




```r
tree_prediction <- predict(tree_fit, newdata=test_set)
MSE <- mean((tree_prediction - test_set$Sales)^2)
cat("The test MSE is:", MSE)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = training_set)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "CompPrice"   "Age"         "Advertising"
## Number of terminal nodes:  15
## Residual mean deviance:  2.608 = 795.6 / 305
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4.15700 -1.07300  0.03188  0.00000  0.93790  3.95200
## The test MSE is: 5.089493
```

(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

`cv.tree()` function is employed to determine the optimal level of tree complexity based on the model from last question. The pruning result is printed after the code. Interestingly, after repeating the pruning process with random training sets and test sets, it is found pruning does not necessarily improve test MSE. While in some cases the pruning process reduces the tree size, in many cases it doesn't, and therefore the test MSE is the same with the previous question.

```r
cv_tree_fit <- cv.tree(tree_fit, FUN=prune.tree, K=10)
cv_tree_fit

cat("The optimal level of tree complexity is determined by cross-validation as:", cv_tree_fit$size[whic

prune_fit <- prune.tree(tree_fit, best=cv_tree_fit$size[which.min(cv_tree_fit$dev)])

prune_prediction <- predict(prune_fit, newdata=test_set)

MSE_prune <- mean((prune_prediction - test_set$Sales)^2)
cat("The test MSE after pruning the tree is:", MSE_prune)
```

```
## $size
##  [1] 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

```
##
## $dev
##  [1] 1478.051 1539.621 1563.488 1620.099 1613.933 1620.350 1658.811
##  [8] 1658.811 1637.725 1600.261 1647.584 1816.467 1837.840 1922.270
## [15] 2554.802
##
## $k
##  [1]      -Inf  31.31824  33.55590  41.53634  42.36674  46.27816  58.09398
##  [8]  58.48234  63.12202  73.91811  85.72281 115.86637 129.99132 258.57519
## [15] 676.69384
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
## The optimal level of tree complexity is determined by cross-validation as: 15The test MSE after prun
```

(d) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important.

Bagging is a special case of Random Forest where for each tree all the predictors will be used. Therefore its implementation is the same with RF. Below is the result of building a Bagging model on the training data and test on the test data. By calling `importance()` function on the fitted model, the importance of variables are printed, where there are two columns, and for each column, a larger value indicates higher importance of the corresponding predictor. In this case, the importance result agrees with previous judgement that `ShelveLoc` and `Price` are the two most important predictors, whereas `CompPrice` is also an influential factor.

```r
set.seed(3)
bagging_fit <- randomForest(Sales~., data=training_set, mtry=ncol(training_set)-1, importance=TRUE)
print(importance(bagging_fit))
prediction_bagging <- predict(bagging_fit, newdata=test_set)
MSE_bagging <- mean((prediction_bagging-test_set$Sales)^2)
cat("The test MSE using Bagging is:", MSE_bagging)
```

```
##                %IncMSE IncNodePurity
## CompPrice   32.341807     260.90909
## Income       6.348797     125.21052
## Advertising 22.235026     176.81412
## Population  -2.507804      73.64554
## Price       72.476126     711.61984
## ShelveLoc   77.479593     830.73574
## Age         18.935520     184.70002
## Education    4.597456      60.91401
## Urban       -2.966190      10.63764
## US           3.332396      14.74628
## The test MSE using Bagging is: 2.441834
```

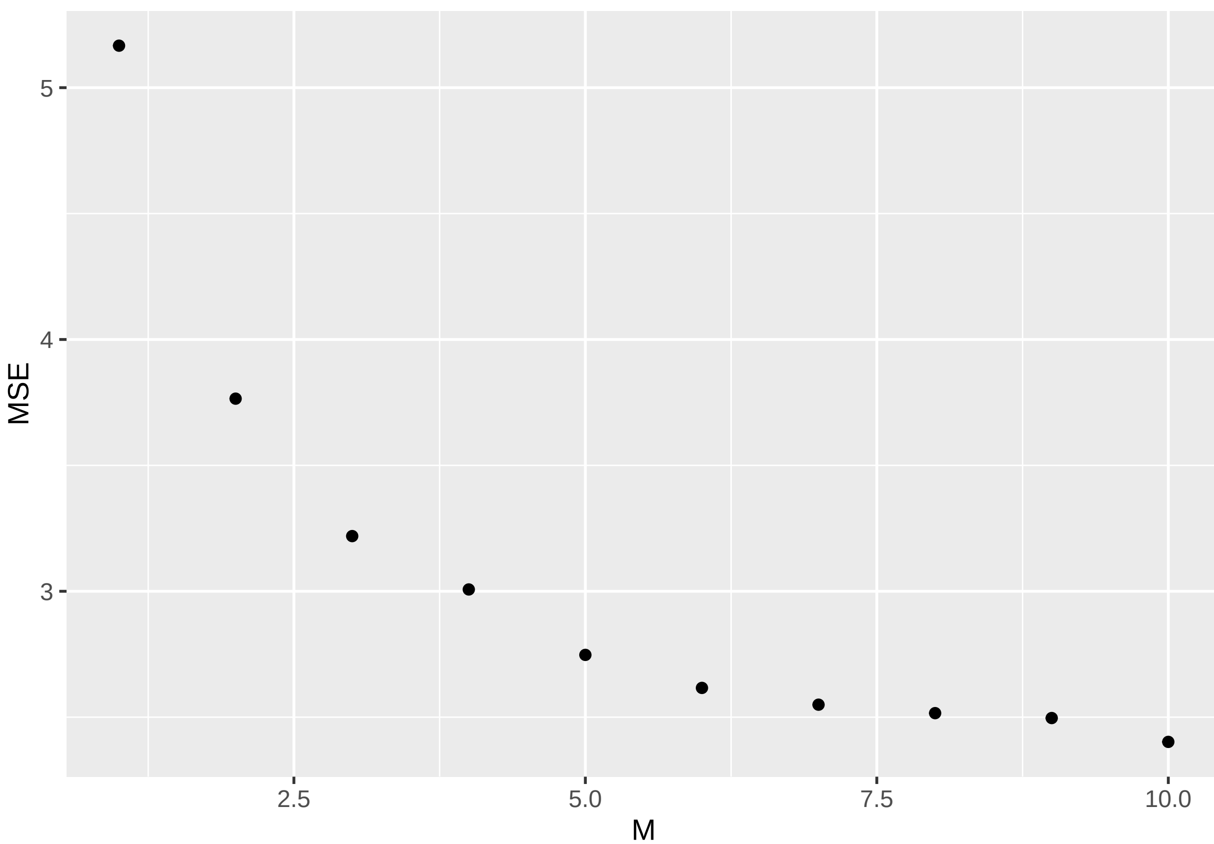
(e) Use random forests to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

To investigate the effect of m on the error rate (also measured as MSE) obtained, a function is defined that takes an m value and produce a RF model on the training data, and test on the test data. The importance of each predictor during the modeling process is also printed. Each prediction's MSE is recorded and plotted against m.

The code output of `importance()` function shows that while `ShelveLoc` remains the most important predictor regardless of m, the relative importance between these variables is more differentiated as m increase, which is saying when m is small, many predictors are relatively important, as they each contains some information about `Sales`, however, when m increases, the less important predictors are overshadowed by the more influential ones.

Looking at the final plot of *MSE vs m*, it is clear that as m becomes larger, the test MSE is reduced, which is equivalent to more accurate predictions.

```
set.seed(4)

rf_m <- function(m){
    rf_fit_tmp <- randomForest(Sales~., data=training_set, ntree=bagging_fit$ntree, mtry=m, importance=
    cat("The variable importance for m =", m, ":")
    print(importance(rf_fit_tmp))
    prediction_rf_tmp <- predict(rf_fit_tmp, newdata=test_set)
    MSE_rf_tmp <- mean((prediction_rf_tmp-test_set$Sales)^2)
    return(MSE_rf_tmp)
}

MSE_rf <- sapply(1:(ncol(training_set)-1), rf_m)

MSE_rf_df <- data.frame(M=1:(ncol(training_set)-1), MSE=MSE_rf)

print(MSE_rf_df)

qplot(x=M, y=MSE, data=MSE_rf_df, geom="auto")
```



```
## The variable importance for m = 1 :              %IncMSE IncNodePurity
## CompPrice    5.429779     144.35721
## Income       2.420363     149.51860
## Advertising 11.264092     155.39130
## Population  -1.403109     148.92287
## Price       22.774688     327.73181
## ShelveLoc   25.584253     327.61444
## Age         11.438369     186.30759
## Education    1.441128     107.27900
## Urban       -1.697863      25.72633
## US           7.179637      56.30667
## The variable importance for m = 2 :              %IncMSE IncNodePurity
## CompPrice   12.743634     208.71427
## Income       1.996174     195.91776
## Advertising 15.975140     209.92284
## Population  -2.702479     175.56093
## Price       36.823191     520.88696
## ShelveLoc   44.147223     535.36344
## Age         14.051359     259.80432
## Education    2.996583     118.49414
## Urban       -2.251717      28.16595
## US           5.811618      54.88950
## The variable importance for m = 3 :               %IncMSE IncNodePurity
## CompPrice   16.8391755     216.42894
## Income       3.7519786     178.82389
## Advertising 17.1648422     221.94309
## Population  -0.5137124     152.43910
## Price       44.7820303     581.38505
## ShelveLoc   53.5715825     618.10366
## Age         16.5440033     261.12001
## Education    1.1383883     101.64083
## Urban       -2.8578171      18.18567
## US           6.1742379      39.92849
## The variable importance for m = 4 :               %IncMSE IncNodePurity
## CompPrice   18.3363502     214.56859
## Income       5.0478246     169.25559
## Advertising 18.9137558     211.49359
## Population  -0.5080351     131.95622
## Price       50.4174167     598.96616
## ShelveLoc   60.0168045     679.95848
## Age         17.3666958     244.15080
## Education    2.2748251      87.01720
## Urban       -2.6735127      14.99929
## US           4.2249724      36.25667
## The variable importance for m = 5 :              %IncMSE IncNodePurity
## CompPrice   22.577706     227.12882
## Income       3.354851     152.07442
## Advertising 19.054906     206.59407
## Population  -1.403651     114.01409
## Price       58.036461     653.83176
## ShelveLoc   69.355616     729.46595
## Age         16.257283     229.61531
## Education    3.255563      76.69778
## Urban       -1.951397      12.82010
```

```
## US           5.372637      28.37526
## The variable importance for m = 6 :                %IncMSE IncNodePurity
## CompPrice   26.2233633     230.20691
## Income       6.4575596     146.64927
## Advertising 21.1193777     203.85468
## Population  -0.7352083      97.08004
## Price       64.4421009     673.45522
## ShelveLoc   71.1831717     751.72061
## Age         17.5983384     215.03580
## Education    2.3621503      73.23473
## Urban       -0.7629621      12.64122
## US           5.7545043      24.28373
## The variable importance for m = 7 :                %IncMSE IncNodePurity
## CompPrice   27.5854437     240.03843
## Income       6.7003899     138.45695
## Advertising 19.5788652     198.66565
## Population  -2.0872826      89.67704
## Price       67.8983348     685.68961
## ShelveLoc   75.0603007     777.08986
## Age         19.0172890     204.05078
## Education    2.1462890      69.28268
## Urban       -0.3447215      11.25105
## US           7.1289122      22.60248
## The variable importance for m = 8 :                %IncMSE IncNodePurity
## CompPrice   30.9792135     248.79112
## Income       7.1783149     135.39630
## Advertising 20.4695965     191.25626
## Population  -0.6009614      83.40223
## Price       68.4059690     706.07189
## ShelveLoc   81.4728695     799.15634
## Age         19.1842568     198.66951
## Education    1.8520415      64.30046
## Urban       -2.7526695      12.16206
## US           6.3383580      17.13218
## The variable importance for m = 9 :                %IncMSE IncNodePurity
## CompPrice   30.8961947     254.23234
## Income       7.2277271     130.62845
## Advertising 22.2917015     189.40969
## Population  -1.2158646      80.17957
## Price       64.8614093     715.73375
## ShelveLoc   81.2373825     803.48074
## Age         19.7676062     192.63684
## Education    0.9220955      59.59559
## Urban       -2.4147137      11.25139
## US           4.3054155      15.71293
## The variable importance for m = 10 :               %IncMSE IncNodePurity
## CompPrice   30.528351     256.58522
## Income       6.457936     129.77315
## Advertising 19.566828     185.73212
## Population  -3.522319      74.99813
## Price       68.064298     717.01386
## ShelveLoc   84.985737     824.01052
## Age         17.995522     184.72320
## Education    2.380724      62.10113
```

```
## Urban       -1.036463      11.05087
## US           3.198280      13.56033
##     M      MSE
## 1   1 5.166777
## 2   2 3.764958
## 3   3 3.219218
## 4   4 3.007261
## 5   5 2.747352
## 6   6 2.616262
## 7   7 2.549604
## 8   8 2.516082
## 9   9 2.496629
## 10 10 2.401967
## Note: no visible global function definition for '.'
```

Problem 11

(a) Create a training set consisting of the first 1,000 observations, and a test set consisting of the remaining observations.

First load `gbm` library.

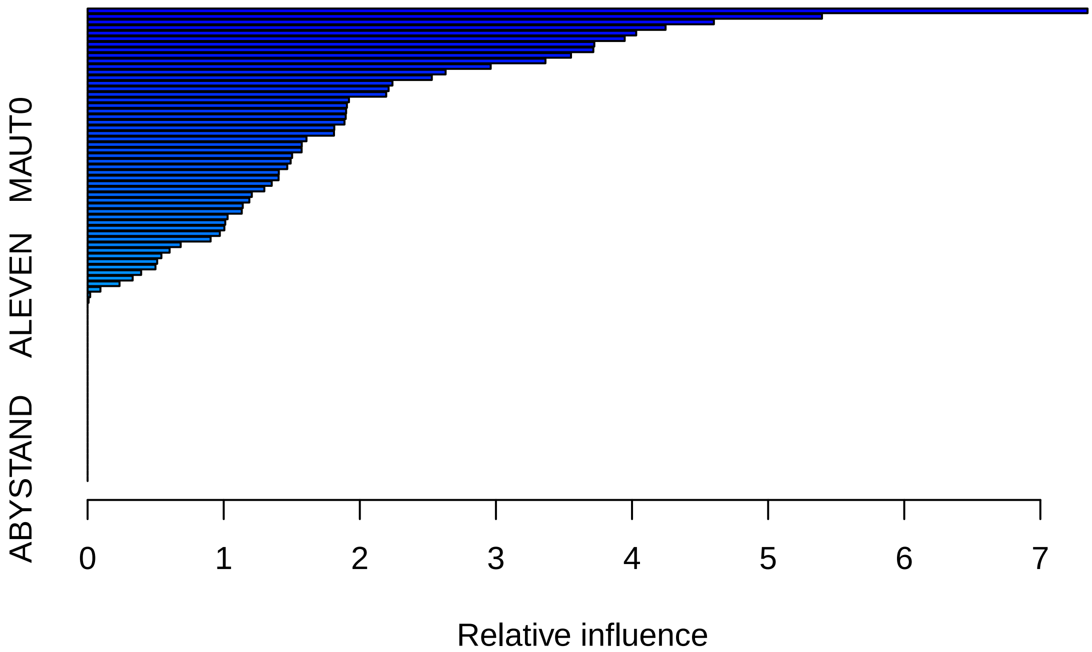
```r
library(gbm)
```

```r
training_set <- Caravan[1:1000,]
test_set <- Caravan[1001:nrow(Caravan),]
```

(b) Fit a boosting model to the training set with Purchase as the response and the other variables as predictors. Use 1,000 trees, and a shrinkage value of 0.01. Which predictors appear to be the most important?

`gbm` function is called to build a boosting model, with 10 fold cross-validation. Here the response is the customer's decision *Yes/No*, which is converted to 1/0, whose distribution is assumed Bernoulli. The predictor relative influence is summarised and plotted by calling `summary()` function.

```r
set.seed(5)
training_set$Purchase <- ifelse(training_set$Purchase=="Yes", 1, 0)
boost_fit <- gbm(Purchase~., data=training_set, n.trees=1000, shrinkage=0.01, interaction.depth=4, dist
print(summary(boost_fit))
```




```
##                 var    rel.inf
## PPERSAUT PPERSAUT 7.34680325
## MGODGE     MGODGE 5.39548712
## PBRAND     PBRAND 4.60149712
## MOPLHOOG MOPLHOOG 4.24655097
## MKOOPKLA MKOOPKLA 4.03062478
## MBERMIDD MBERMIDD 3.94614542
## MGODPR     MGODPR 3.72290610
## MAUT2       MAUT2 3.71556936
## MOSTYPE   MOSTYPE 3.55166504
## MINK3045 MINK3045 3.36401838
## MBERARBG MBERARBG 2.96171305
## MSKC         MSKC 2.63038874
## MINK7512 MINK7512 2.52914683
## MOPLMIDD MOPLMIDD 2.23959842
## MSKB1       MSKB1 2.21174140
## MSKA         MSKA 2.19404408
## MBERARBO MBERARBO 1.92004167
## MFGEKIND MFGEKIND 1.90520749
## MFWEKIND MFWEKIND 1.89947463
## MINKM30   MINKM30 1.89632539
## MAUT1       MAUT1 1.88732036
## MBERHOOG MBERHOOG 1.81227127
## PWAPART   PWAPART 1.81026021
## MRELOV     MRELOV 1.60817182
## MGODOV     MGODOV 1.57359406
## MAUT0       MAUT0 1.57249606
## MRELGE     MRELGE 1.50254531
## MSKB2       MSKB2 1.49196545
## MFALLEEN MFALLEEN 1.46695268
## MHKOOP     MHKOOP 1.40487687
## MGODRK     MGODRK 1.40324377
## MHHUUR     MHHUUR 1.35281406
```

```
## MZFONDS   MZFONDS 1.29851341
## MINKGEM   MINKGEM 1.20648592
## MGEMLEEF MGEMLEEF 1.18837952
## ABRAND     ABRAND 1.13999047
## MINK4575 MINK4575 1.13264166
## MSKD         MSKD 1.02862928
## MZPART     MZPART 1.01145611
## MRELSA     MRELSA 1.00471332
## MGEMOMV   MGEMOMV 0.97160656
## APERSAUT APERSAUT 0.90379002
## MOSHOOFD MOSHOOFD 0.68392032
## MOPLLAAG MOPLLAAG 0.60265978
## PMOTSCO   PMOTSCO 0.54207708
## MBERZELF MBERZELF 0.51124136
## MBERBOER MBERBOER 0.49862579
## MINK123M MINK123M 0.39351160
## PLEVEN     PLEVEN 0.33116978
## PBYSTAND PBYSTAND 0.23470646
## MAANTHUI MAANTHUI 0.09417492
## ALEVEN     ALEVEN 0.01888435
## PAANHANG PAANHANG 0.00736115
## PWABEDR   PWABEDR 0.00000000
## PWALAND   PWALAND 0.00000000
## PBESAUT   PBESAUT 0.00000000
## PVRAAUT   PVRAAUT 0.00000000
## PTRACTOR PTRACTOR 0.00000000
## PWERKT     PWERKT 0.00000000
## PBROM       PBROM 0.00000000
## PPERSONG PPERSONG 0.00000000
## PGEZONG   PGEZONG 0.00000000
## PWAOREG   PWAOREG 0.00000000
## PZEILPL   PZEILPL 0.00000000
## PPLEZIER PPLEZIER 0.00000000
## PFIETS     PFIETS 0.00000000
## PINBOED   PINBOED 0.00000000
## AWAPART   AWAPART 0.00000000
## AWABEDR   AWABEDR 0.00000000
## AWALAND   AWALAND 0.00000000
## ABESAUT   ABESAUT 0.00000000
## AMOTSCO   AMOTSCO 0.00000000
## AVRAAUT   AVRAAUT 0.00000000
## AAANHANG AAANHANG 0.00000000
## ATRACTOR ATRACTOR 0.00000000
## AWERKT     AWERKT 0.00000000
## ABROM       ABROM 0.00000000
## APERSONG APERSONG 0.00000000
## AGEZONG   AGEZONG 0.00000000
## AWAOREG   AWAOREG 0.00000000
## AZEILPL   AZEILPL 0.00000000
## APLEZIER APLEZIER 0.00000000
## AFIETS     AFIETS 0.00000000
## AINBOED   AINBOED 0.00000000
## ABYSTAND ABYSTAND 0.00000000
```

(c) Use the boosting model to predict the response on the test data. Predict that a person will make a purchase if the estimated probability of purchase is greater than 20 %. Form a confusion matrix. What fraction of the people predicted to make a purchase do in fact make one? How does this compare with the results obtained from applying KNN or logistic regression to this data set?

The confusion matrix using boosting on the test data is tabulated below.

```
prediction_boost <- sapply(predict(boost_fit, newdata=test_set, type="response", n.trees=1000), functio
tabl <- table(prediction_boost, test_set$Purchase)
tabl
cat(100*tabl[2,2]/sum(tabl[2,]), "% of the predicted purchases happened in reality.")
```

```
##
## prediction_boost   No  Yes
##              No  4343  259
##              Yes  190   30
## 13.63636 % of the predicted purchases happened in reality.
```

Generally logistic regression is more suitable for predicting binomial variables, and therefore it is compared with boosting. Here the decision threshold for a purchase decision is the same with boosting as 20%.

```
set.seed(6)
logistic_fit <- glm(Purchase~., data=training_set, family=binomial)
prediction_logistic <- sapply(predict(logistic_fit, newdata=test_set, type="response"), function(x) ifel
tabl1 <- table(prediction_logistic, test_set$Purchase)
tabl1
cat(100*tabl1[2,2]/sum(tabl1[2,]), "% of the predicted purchases happened in reality (logistic regressi
```

```
##
## prediction_logistic   No  Yes
##                 No  4183  231
##                 Yes  350   58
## 14.21569 % of the predicted purchases happened in reality (logistic regression).
```

From the output it is seen that logistic regression's prediction has a slightly higher purchase conversion rate compared with boosting, however, in terms of correct prediction, i.e. both correct *Yes* and *No*, boosting is slightly higher than logistic regression.