hw5_shuangyu_zhao

shuangyu_zhao

2023-03-14

```
library(ISLR2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tree)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(e1071)
```

1

```
default_df <- Default
glimpse(default_df)
```

```
## Rows: 10,000
## Columns: 4
## $ default <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, No, No, No~
## $ student <fct> No, Yes, No, No, No, Yes, No, Yes, No, No, Yes, Yes, No, No, N~
## $ balance <dbl> 729.5265, 817.1804, 1073.5492, 529.2506, 785.6559, 919.5885, 8~
## $ income  <dbl> 44361.625, 12106.135, 31767.139, 35704.494, 38463.496, 7491.55~
```

(a) using glm() and summary() to get the 95% confidence interval of parameters

```r
split_pct <- 0.7
n <- length(default_df$default)*split_pct # train size
row_samp <- sample(1:length(default_df$default), n, replace = FALSE)
train1 <- default_df[row_samp,]
test1 <- default_df[-row_samp,]
model1 <- glm(data = train1, default ~ balance + income, family = binomial)
summary(model1)
```

```
##
## Call:
## glm(formula = default ~ balance + income, family = binomial,
##     data = train1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5379  -0.1327  -0.0500  -0.0174   3.3820
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.189e+01  5.457e-01 -21.797   <2e-16 ***
## balance      5.974e-03  2.900e-04  20.598   <2e-16 ***
## income       1.569e-05  6.084e-06   2.579   0.0099 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2043.8  on 6999  degrees of freedom
## Residual deviance: 1056.4  on 6997  degrees of freedom
## AIC: 1062.4
##
## Number of Fisher Scoring iterations: 8
```

the 95% CI of parameter:

```r
para_bal <- summary(model1)$coef[2, 1]
para_inc <- summary(model1)$coef[3, 1]
se_bal <- summary(model1)$coef[2, 2]
se_inc <- summary(model1)$coef[3, 2]
```

```r
CI_bal <- para_bal + se_bal * qt(c(0.025, 0.975), n-2)
print(paste0("the confidence of interval of balance's parameter: ", CI_bal[1], '~',CI_bal[2]))
```

```
## [1] "the confidence of interval of balance's parameter: 0.00540585917076432~0.00654303018400874"
```

```r
CI_inc <- para_inc + se_inc * qt(c(0.025, 0.975), n-2)
print(paste0("the confidence of interval of income's parameter: ", CI_inc[1], '~',CI_inc[2]))
```

```
## [1] "the confidence of interval of income's parameter: 3.76667422070801e-06~2.7619246145388e-05"
```

(b) get the 95% confidence interval of parameters by bootstrapping

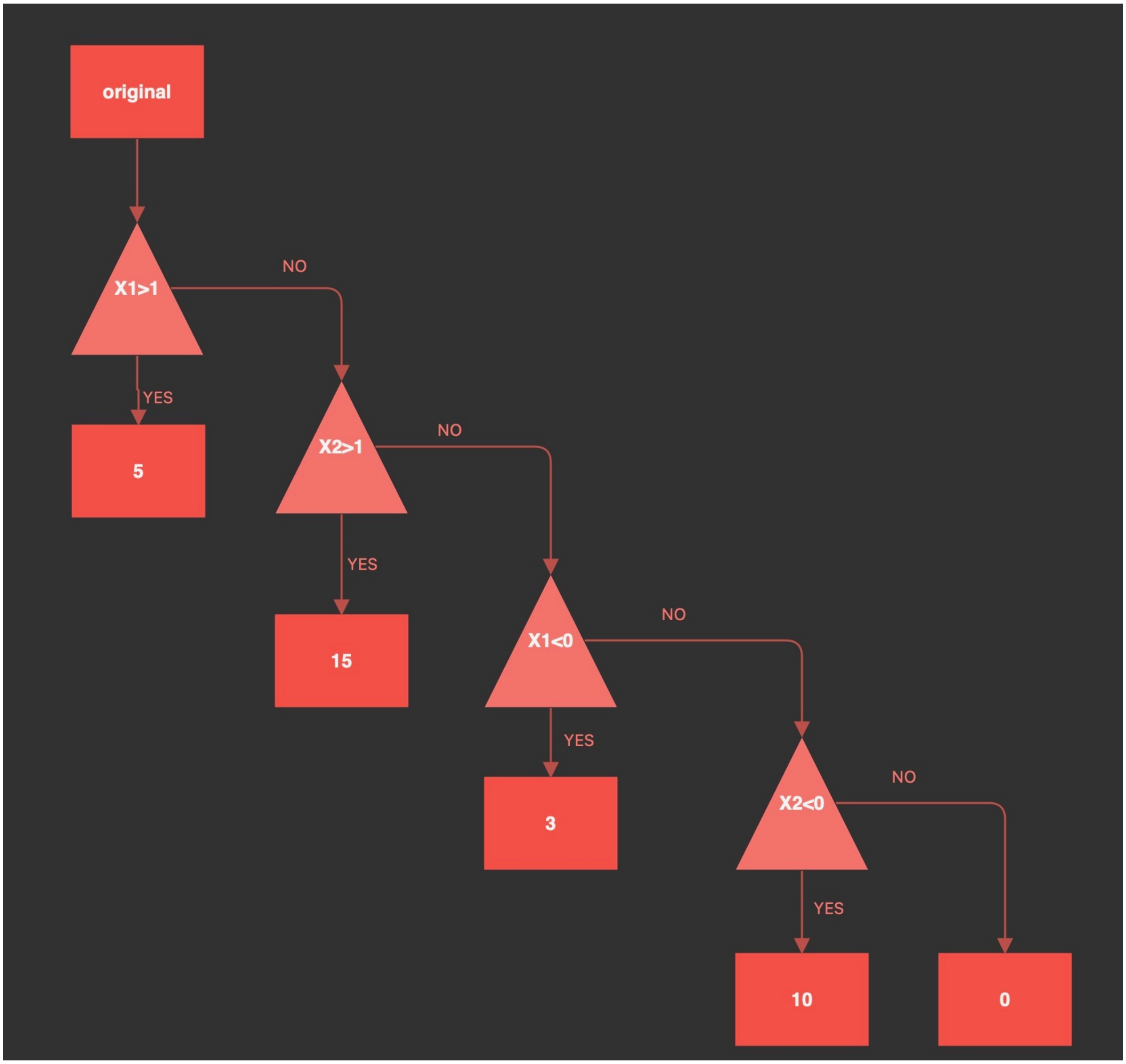
```r
coeff_bal <- rep(0, 1000)
coeff_inc <- rep(0, 1000)
n <- nrow(default_df)
for(i in 1:1000){
  row_samp <- sample(1:n, replace = TRUE)
  samp <- default_df[row_samp,]
  model2 <- glm(data = samp, default ~ balance + income, family = binomial)
  coeff_bal[i] <- model2$coefficients[2]
  coeff_inc[i] <- model2$coefficients[3]
}
```

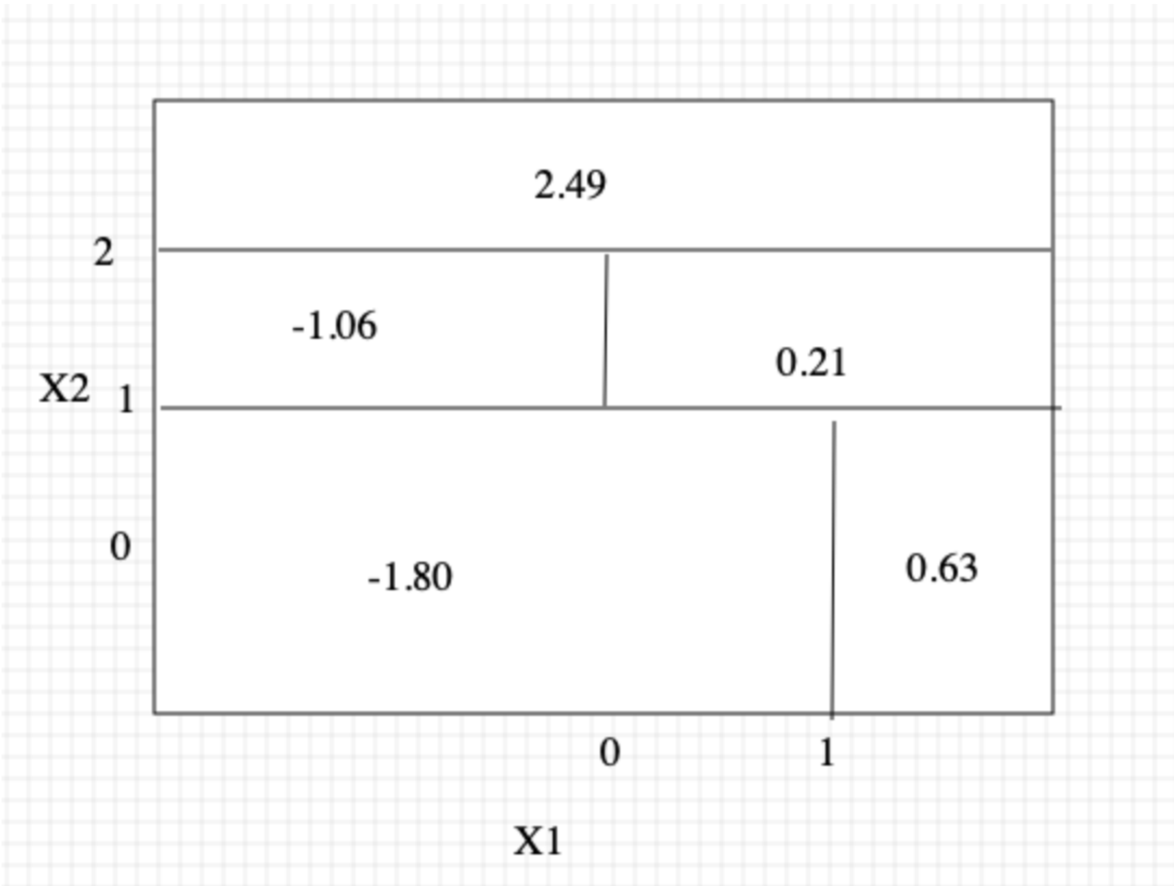
```r
print(paste0("the confidence of interval of balance's parameter: ", quantile(coeff_bal, 0.025), '~', qua
```

```
## [1] "the confidence of interval of balance's parameter: 0.00523387178640283~0.00611320442021624"
```

```r
print(paste0("the confidence of interval of income's parameter: ", quantile(coeff_inc, 0.025), '~', quan
```

```
## [1] "the confidence of interval of income's parameter: 1.15551675063052e-05~3.0102170886149e-05"
```

2 (a) 

(b) 

3 (a)

```
set.seed(123)
row_num <- sample(1:length(OJ$Purchase), 800, replace = FALSE)
train_3 <- OJ[row_num, ]
test_3 <- OJ[-row_num, ]
```

(b)

```
glimpse(train_3)
```

```
## Rows: 800
## Columns: 18
## $ Purchase       <fct> MM, CH, CH, MM, CH, MM, MM, MM, CH, MM, CH, MM, CH, CH,~
## $ WeekofPurchase <dbl> 238, 228, 244, 263, 271, 233, 233, 228, 261, 259, 231, ~
## $ StoreID        <dbl> 3, 7, 7, 1, 4, 2, 1, 2, 7, 3, 7, 2, 7, 7, 1, 7, 2, 7, 4~
## $ PriceCH        <dbl> 1.79, 1.69, 1.86, 1.76, 1.99, 1.69, 1.69, 1.69, 1.86, 1~
## $ PriceMM        <dbl> 2.09, 1.69, 2.09, 1.99, 2.09, 1.69, 1.99, 1.69, 2.13, 2~
## $ DiscCH         <dbl> 0.00, 0.00, 0.00, 0.00, 0.10, 0.00, 0.00, 0.00, 0.00, 0~
## $ DiscMM         <dbl> 0.00, 0.00, 0.20, 0.40, 0.40, 0.00, 0.00, 0.00, 0.24, 0~
## $ SpecialCH      <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0~
## $ SpecialMM      <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1~
## $ LoyalCH        <dbl> 0.145350, 0.584000, 0.992794, 0.400000, 0.999797, 0.108~
```

```
## $ SalePriceMM    <dbl> 2.09, 1.69, 1.89, 1.59, 1.69, 1.69, 1.99, 1.69, 1.89, 2~
## $ SalePriceCH    <dbl> 1.79, 1.69, 1.86, 1.76, 1.89, 1.69, 1.69, 1.69, 1.86, 1~
## $ PriceDiff      <dbl> 0.30, 0.00, 0.03, -0.17, -0.20, 0.00, 0.30, 0.00, 0.03,~
## $ Store7         <fct> No, Yes, Yes, No, No, No, No, No, Yes, No, Yes, No, Yes~
## $ PctDiscMM      <dbl> 0.000000, 0.000000, 0.095694, 0.201005, 0.191388, 0.000~
## $ PctDiscCH      <dbl> 0.000000, 0.000000, 0.000000, 0.000000, 0.050251, 0.000~
## $ ListPriceDiff  <dbl> 0.30, 0.00, 0.23, 0.23, 0.10, 0.00, 0.30, 0.00, 0.27, 0~
## $ STORE          <dbl> 3, 0, 0, 1, 4, 2, 1, 2, 0, 3, 0, 2, 0, 0, 1, 0, 2, 0, 4~
```

```r
model3 <- tree(data = train_3, Purchase ~ WeekofPurchase + StoreID + PriceCH + PriceMM + DiscCH + DiscMM
summary(model3)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ WeekofPurchase + StoreID + PriceCH +
##     PriceMM + DiscCH + DiscMM + SpecialCH + SpecialMM + LoyalCH +
##     SalePriceMM + SalePriceCH + PriceDiff + Store7 + PctDiscMM +
##     PctDiscCH + ListPriceDiff + STORE, data = train_3, method = "class")
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7625 = 603.9 / 792
## Misclassification error rate: 0.165 = 132 / 800
```

```r
print(paste0("train error rate: ", 0.165))
```

```
## [1] "train error rate: 0.165"
```

and this model has 8 terminal nodes

(c)

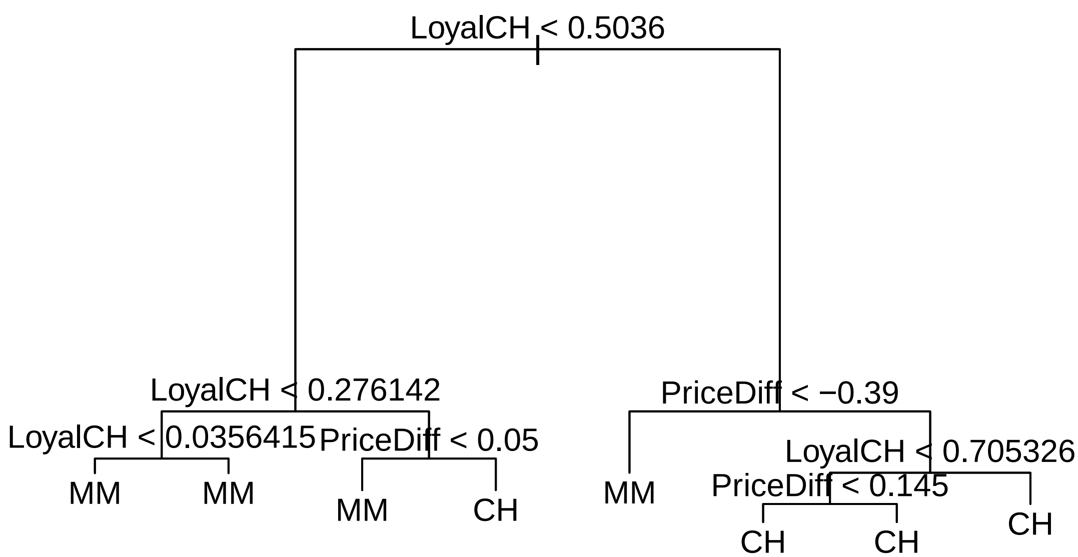
```r
model3
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1071.00 CH ( 0.60875 0.39125 )
##    2) LoyalCH < 0.5036 350  415.10 MM ( 0.28000 0.72000 )
##      4) LoyalCH < 0.276142 170  131.00 MM ( 0.12941 0.87059 )
##        8) LoyalCH < 0.0356415 56   10.03 MM ( 0.01786 0.98214 ) *
##        9) LoyalCH > 0.0356415 114  108.90 MM ( 0.18421 0.81579 ) *
##      5) LoyalCH > 0.276142 180  245.20 MM ( 0.42222 0.57778 )
##       10) PriceDiff < 0.05 74   74.61 MM ( 0.20270 0.79730 ) *
##       11) PriceDiff > 0.05 106  144.50 CH ( 0.57547 0.42453 ) *
##    3) LoyalCH > 0.5036 450  357.10 CH ( 0.86444 0.13556 )
##      6) PriceDiff < -0.39 27   32.82 MM ( 0.29630 0.70370 ) *
##      7) PriceDiff > -0.39 423  273.70 CH ( 0.90071 0.09929 )
##       14) LoyalCH < 0.705326 130  135.50 CH ( 0.78462 0.21538 )
##         28) PriceDiff < 0.145 43   58.47 CH ( 0.58140 0.41860 ) *
##         29) PriceDiff > 0.145 87   62.07 CH ( 0.88506 0.11494 ) *
##       15) LoyalCH > 0.705326 293  112.50 CH ( 0.95222 0.04778 ) *
```

if the rows follow the rule in 2), then go to 4) or 5), or follow the rule in 3), then goto 6) or 7)

(d)

```
plot(model3)
text(model3)
```



if yes, go left. if not, go right.

(e)

```
predict3 <- predict(model3, test_3, type = 'class')
confusionMatrix(data = predict3, reference = as.factor(test_3$Purchase))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 150  34
##         MM  16  70
##
##                Accuracy : 0.8148
##                  95% CI : (0.7633, 0.8593)
##     No Information Rate : 0.6148
##     P-Value [Acc > NIR] : 9.56e-13
```

```
##
##                  Kappa : 0.596
##
##  Mcnemar's Test P-Value : 0.01621
##
##            Sensitivity : 0.9036
##            Specificity : 0.6731
##         Pos Pred Value : 0.8152
##         Neg Pred Value : 0.8140
##             Prevalence : 0.6148
##         Detection Rate : 0.5556
##   Detection Prevalence : 0.6815
##      Balanced Accuracy : 0.7883
##
##       'Positive' Class : CH
##
```

```
print(paste0("the test error rate: ", 1-confusionMatrix(data = predict3, reference = as.factor(test_3$Pu
```

```
## [1] "the test error rate: 0.185185185185185"
```

4 (a) logistic regression model

```
model4a <- glm(data = train_3, Purchase ~ PriceDiff + LoyalCH, family = binomial)
```

(b) naive bayes model

```
model4b <- naiveBayes(data = train_3, Purchase ~ PriceDiff + LoyalCH)
```

(c) decision tree model

```
model4c <- tree(data = train_3, Purchase ~ PriceDiff + LoyalCH, method = "class")
```

(d) esemble the prediction

```
predict4a <- predict(model4a, test_3, type = 'response')
predicted_classes_4a <- ifelse(predict4a > 0.5, "MM", "CH")
predict4b <- predict(model4b, test_3)
predict4c <- predict(model4c, test_3, type = 'class')
```

```
esemble_predict <- rep(0, length(predict4a))
for(i in 1:length(predict4a)){
  if ((predicted_classes_4a[i] == "CH") + (predict4b[i] == "CH") + (predict4c[i] == "CH") >= 2 ){
    esemble_predict[i] <- "CH"
  }else{
    esemble_predict[i] <- "MM"
  }
}
```

```r
# confusion matrix for logistic regression model
confusionMatrix(data = as.factor(predicted_classes_4a), reference = as.factor(test_3$Purchase))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 146  25
##         MM  20  79
##
##                Accuracy : 0.8333
##                  95% CI : (0.7834, 0.8758)
##     No Information Rate : 0.6148
##     P-Value [Acc > NIR] : 4.285e-15
##
##                   Kappa : 0.6449
##
##  Mcnemar's Test P-Value : 0.551
##
##             Sensitivity : 0.8795
##             Specificity : 0.7596
##          Pos Pred Value : 0.8538
##          Neg Pred Value : 0.7980
##              Prevalence : 0.6148
##          Detection Rate : 0.5407
##    Detection Prevalence : 0.6333
##       Balanced Accuracy : 0.8196
##
##        'Positive' Class : CH
##
```

```r
# confusion matrix for naive bayes model
confusionMatrix(data = as.factor(predict4b), reference = as.factor(test_3$Purchase))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 143  27
##         MM  23  77
##
##                Accuracy : 0.8148
##                  95% CI : (0.7633, 0.8593)
##     No Information Rate : 0.6148
##     P-Value [Acc > NIR] : 9.56e-13
##
##                   Kappa : 0.6062
##
##  Mcnemar's Test P-Value : 0.6714
##
##             Sensitivity : 0.8614
##             Specificity : 0.7404
##          Pos Pred Value : 0.8412
```

```
##          Neg Pred Value : 0.7700
##              Prevalence : 0.6148
##          Detection Rate : 0.5296
##    Detection Prevalence : 0.6296
##       Balanced Accuracy : 0.8009
##
##        'Positive' Class : CH
##
```

```r
# confusion matrix for decision tree model
confusionMatrix(data = as.factor(predict4c), reference = as.factor(test_3$Purchase))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 150  34
##         MM  16  70
##
##                Accuracy : 0.8148
##                  95% CI : (0.7633, 0.8593)
##     No Information Rate : 0.6148
##     P-Value [Acc > NIR] : 9.56e-13
##
##                   Kappa : 0.596
##
##  Mcnemar's Test P-Value : 0.01621
##
##             Sensitivity : 0.9036
##             Specificity : 0.6731
##          Pos Pred Value : 0.8152
##          Neg Pred Value : 0.8140
##              Prevalence : 0.6148
##          Detection Rate : 0.5556
##    Detection Prevalence : 0.6815
##       Balanced Accuracy : 0.7883
##
##        'Positive' Class : CH
##
```

```r
# confusion matrix of esembled result
confusionMatrix(data = as.factor(esemble_predict), reference = as.factor(test_3$Purchase))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 146  26
##         MM  20  78
##
##                Accuracy : 0.8296
##                  95% CI : (0.7794, 0.8725)
##     No Information Rate : 0.6148
```

```
##     P-Value [Acc > NIR] : 1.329e-14
##
##                   Kappa : 0.6364
##
##  Mcnemar's Test P-Value : 0.461
##
##             Sensitivity : 0.8795
##             Specificity : 0.7500
##          Pos Pred Value : 0.8488
##          Neg Pred Value : 0.7959
##              Prevalence : 0.6148
##          Detection Rate : 0.5407
##    Detection Prevalence : 0.6370
##       Balanced Accuracy : 0.8148
##
##        'Positive' Class : CH
##
```

logistic regression has highest accuracy. The accuracy of esembled predicted result is in the middle of these results.

5

```r
predict5 <- matrix(nrow = length(test_3$Purchase), ncol = 0)
for(i in 1:1000){
  rows <- sample(1:length(train_3$Purchase), length(train_3$Purchase), replace = TRUE)
  samp <- train_3[rows,]
  trees <- tree(data = samp, Purchase ~ WeekofPurchase + StoreID + PriceCH + PriceMM + DiscCH + DiscMM +
  predict5 <- cbind(predict5, predict(trees, test_3)[,1])
}
ens <- rowMeans(predict5)
```

```r
confusionMatrix(as.factor(ifelse(ens < 0.5, 'MM', 'CH')), reference = test_3$Purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 143  26
##         MM  23  78
##
##                Accuracy : 0.8185
##                  95% CI : (0.7673, 0.8626)
##     No Information Rate : 0.6148
##     P-Value [Acc > NIR] : 3.407e-13
##
##                   Kappa : 0.6148
##
##  Mcnemar's Test P-Value : 0.7751
##
##             Sensitivity : 0.8614
##             Specificity : 0.7500
##          Pos Pred Value : 0.8462
```

```
##          Neg Pred Value : 0.7723
##              Prevalence : 0.6148
##          Detection Rate : 0.5296
##    Detection Prevalence : 0.6259
##       Balanced Accuracy : 0.8057
##
##        'Positive' Class : CH
##
```

accuracy is higher.