machine learning(732A99) lab2

Anubhav Dikshit(anudi287)

10 December 2018

Contents

Assignment 1

Loading The Libraries

Loading Input files

```
credit_data <- read.xlsx("creditscoring.xls", sheetName = "credit")
credit_data$good_bad <- as.factor(credit_data$good_bad)
```

Assignment 2

2.1 Import the data to R and divide into training/validation/test as 50/25/25: use data partitioning code specified in Lecture 1e.

```
set.seed(12345)

n=NROW(credit_data)
set.seed(12345)
id=sample(1:n, floor(n*0.4))
train=credit_data[id,]

id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.3))
valid=credit_data[id2,]

id3=setdiff(id1,id2)
test=credit_data[id3,]
```

2.2 Fit a decision tree to the training data by using the following measures of impurity: a. Deviance b. Gini index and report the misclassification rates for the training and test data. Choose the measure providing the better results for the following steps.

```
# Create a decision tree model
credit_tree_deviance <- tree(good_bad~., data=train, split = c("deviance"))
credit_tree_gini <- tree(good_bad~., data=train, split = c("gini"))

# Visualize the decision tree with rpart.plot
summary(credit_tree_deviance)
```

```
##
## Classification tree:
## tree(formula = good_bad ~ ., data = train, split = c("deviance"))
## Variables actually used in tree construction:
##  [1] "duration" "history"  "marital"  "existcr"  "amount"   "purpose"
##  [7] "savings"  "resident" "age"      "other"
## Number of terminal nodes:  22
```

```
## Residual mean deviance:  0.7423 = 277.6 / 374
## Misclassification error rate: 0.1869 = 74 / 396
```

```
summary(credit_tree_gini)
```

```
##
## Classification tree:
## tree(formula = good_bad ~ ., data = train, split = c("gini"))
## Variables actually used in tree construction:
##  [1] "foreign"  "coapp"    "depends"  "telephon" "existcr"  "savings"
##  [7] "history"  "property" "amount"   "marital"  "duration" "resident"
## [13] "job"      "installp" "purpose"  "employed" "housing"
## Number of terminal nodes:  53
## Residual mean deviance:  0.9468 = 324.7 / 343
## Misclassification error rate: 0.2247 = 89 / 396
```

```
# predicting on the test dataset to get the misclassification rate.
predict_tree_deviance <- predict(credit_tree_deviance, newdata = test, type = "class")
predict_tree_gini <- predict(credit_tree_gini, newdata = test, type = "class")

conf_tree_deviance <- table(test$good_bad, predict_tree_deviance)
names(dimnames(conf_tree_deviance)) <- c("Actual Test", "Predicted Test")
caret::confusionMatrix(conf_tree_deviance)
```

```
## Confusion Matrix and Statistics
##
##            Predicted Test
## Actual Test bad good
##        bad   49   43
##        good  56  152
##
##                Accuracy : 0.67
##                  95% CI : (0.6136, 0.723)
##     No Information Rate : 0.65
##     P-Value [Acc > NIR] : 0.2539
##
##                   Kappa : 0.2534
##  Mcnemar's Test P-Value : 0.2278
##
##             Sensitivity : 0.4667
##             Specificity : 0.7795
##          Pos Pred Value : 0.5326
##          Neg Pred Value : 0.7308
##              Prevalence : 0.3500
##          Detection Rate : 0.1633
##    Detection Prevalence : 0.3067
##       Balanced Accuracy : 0.6231
##
##        'Positive' Class : bad
##
```

```
conf_tree_gini <- table(test$good_bad, predict_tree_gini)
names(dimnames(conf_tree_gini)) <- c("Actual Test", "Predicted Test")
caret::confusionMatrix(conf_tree_gini)
```

```
## Confusion Matrix and Statistics
```

```
##
##            Predicted Test
## Actual Test bad good
##        bad   24   68
##        good  44  164
##
##                Accuracy : 0.6267
##                  95% CI : (0.5692, 0.6816)
##     No Information Rate : 0.7733
##     P-Value [Acc > NIR] : 1.00000
##
##                   Kappa : 0.0532
##  Mcnemar's Test P-Value : 0.02976
##
##             Sensitivity : 0.3529
##             Specificity : 0.7069
##          Pos Pred Value : 0.2609
##          Neg Pred Value : 0.7885
##              Prevalence : 0.2267
##          Detection Rate : 0.0800
##    Detection Prevalence : 0.3067
##       Balanced Accuracy : 0.5299
##
##        'Positive' Class : bad
##
```

Analysis: On the Training dataset model with 'deviance' had a misclassfication rate of 21% while the model with 'gini' split had the misclassification rate of 23.68%.

For the test dataset we see that the model with 'deviance' type of split has a accuracy of 73.4% or misclassifiaction rate of 26.6%, we see that to predict 'good' the accuracy is 89% but for predicting bad its just 37%. Thus our model is heavily baised towards predicting cases as good.

For the test dataset we see that the model with 'gini' type of split has a accuracy of 65.8% or misclassifiaction rate of 34.2%, we see that to predict 'good' the accuracy is 82% but for predicting bad its just 28%. Thus our model is heavily baised towards predicting cases as good.

Both our models would lead to many bad loan applicant to be given loans which is never a good thing, however among the model the one using 'deviance' mode for split is better by 7.6%.

Thus we will select model using 'deviance' for further model building.

3. Use training and validation sets to choose the optimal tree depth. Present the graphs of the dependence of deviances for the training and the validation data on the number of leaves. Report the optimal tree, report it's depth and the variables used by the tree. Interpret the information provided by the tree structure. Estimate the misclassification rate for the test data.

```r
set.seed(12345)


credit_tree <- tree(good_bad~., data=train, split = c("deviance"))

credit_tree_purned_train <- prune.tree(credit_tree, method = c("deviance"))
```

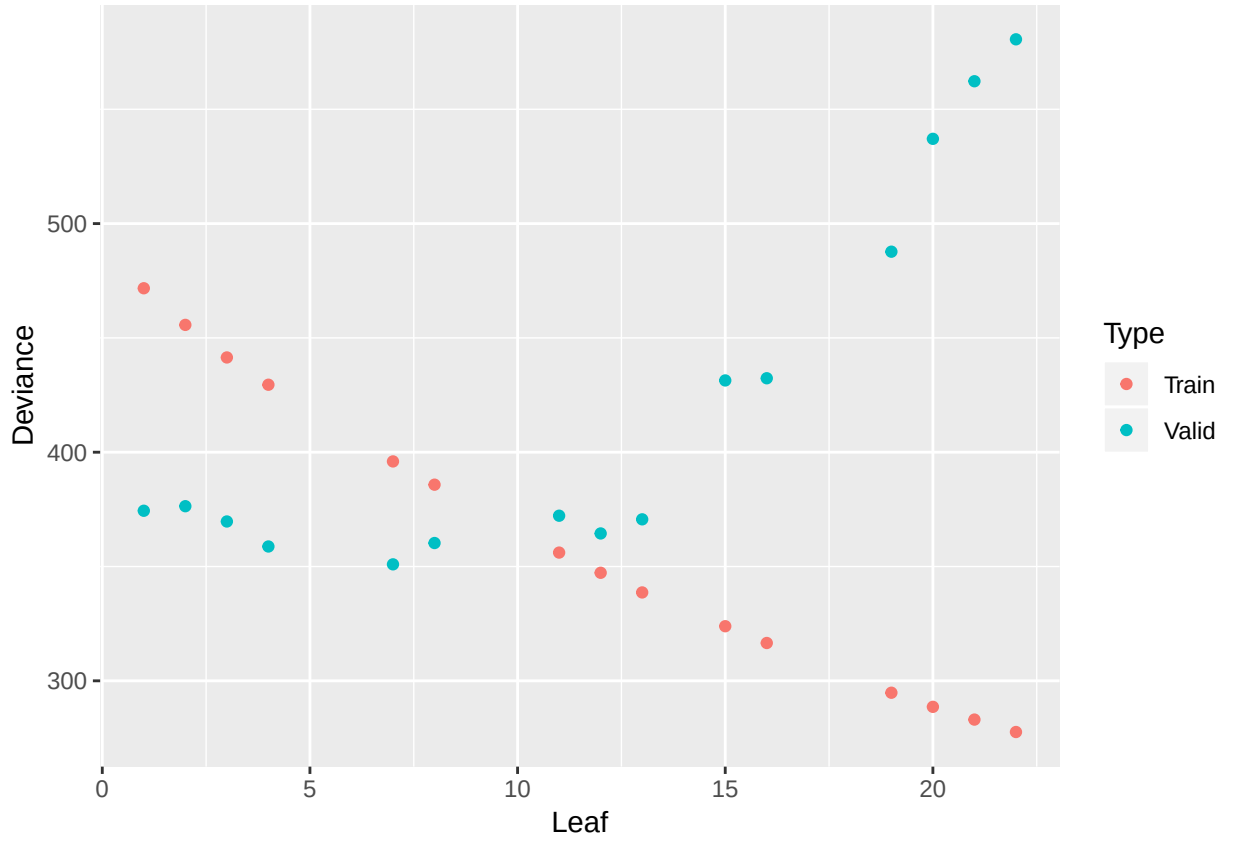
```
credit_tree_purned_test <- prune.tree(credit_tree, newdata = valid ,method = c("deviance"))

result_train <- cbind(credit_tree_purned_train$size, credit_tree_purned_train$dev, "Train")
result_test <- cbind(credit_tree_purned_test$size, credit_tree_purned_test$dev, "Valid")

result <- as.data.frame(rbind(result_test, result_train))
colnames(result) <- c("Leaf", "Deviance", "Type")

result$Leaf <- as.numeric(as.character(result$Leaf))
result$Deviance <- as.numeric(as.character(result$Deviance))


ggplot(data = result, aes(x = Leaf, y = Deviance, colour = Type)) + geom_point()
```



Analysis:

4. Use training data to perform classification using Naïve Bayes and report the confusion matrices and misclassification rates for the training and for the test data. Compare the results with those from step 3.

```
#Fitting the Naive Bayes model
credit_naive_model = naiveBayes(good_bad ~., data=train)

credit_naive_model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##   bad  good
## 0.285 0.715
##
## Conditional probabilities:
##       duration
## Y          [,1]     [,2]
##   bad  24.85965 14.95946
##   good 19.13287 11.11076
##
##       history
## Y          [,1]     [,2]
##   bad  2.263158 1.113494
##   good 2.744755 1.033419
##
##       purpose
## Y          [,1]     [,2]
##   bad  2.883929 2.859300
##   good 2.718310 2.409833
##
##       amount
## Y          [,1]     [,2]
##   bad  3661.202 3504.882
##   good 2995.524 2431.313
##
##       savings
## Y          [,1]     [,2]
##   bad  1.736842 1.343965
##   good 2.314685 1.664572
##
##       employed
## Y          [,1]     [,2]
##   bad  3.236842 1.250151
##   good 3.500000 1.147843
##
##       installp
## Y          [,1]      [,2]
##   bad  3.175439 0.9798466
##   good 2.898601 1.1271131
##
##       marital
## Y          [,1]      [,2]
##   bad  2.596491 0.7252622
##   good 2.716783 0.6650668
##
##       coapp
## Y          [,1]      [,2]
```

```
##   bad  1.105263 0.4072011
##   good 1.171329 0.5383695
##
##       resident
## Y          [,1]     [,2]
##   bad  2.824561 1.074611
##   good 2.807692 1.121439
##
##       property
## Y          [,1]     [,2]
##   bad  2.456140 1.073744
##   good 2.300699 1.069743
##
##       age
## Y          [,1]     [,2]
##   bad  33.91228 10.97589
##   good 36.51399 11.21050
##
##       other
## Y          [,1]      [,2]
##   bad  2.500000 0.8227947
##   good 2.751748 0.6362558
##
##       housing
## Y          [,1]      [,2]
##   bad  1.938596 0.5991895
##   good 1.965035 0.5014028
##
##       existcr
## Y          [,1]      [,2]
##   bad  1.368421 0.5988656
##   good 1.458042 0.5893693
##
##       job
## Y          [,1]      [,2]
##   bad  2.824561 0.7194592
##   good 2.849650 0.6503305
##
##       depends
## Y          [,1]      [,2]
##   bad  1.140351 0.3488843
##   good 1.164336 0.3712295
##
##       telephon
## Y          [,1]      [,2]
##   bad  1.315789 0.4668818
##   good 1.388112 0.4881745
##
##       foreign
## Y          [,1]      [,2]
##   bad  1.017544 0.1318659
##   good 1.045455 0.2086640
```

```r
#Prediction on the dataset
predict_naive_train = predict(credit_naive_model, newdata=train)
predict_naive_test = predict(credit_naive_model, newdata=test)


conf_naive_train <- table(train$good_bad, predict_naive_train)
names(dimnames(conf_naive_train)) <- c("Actual Train", "Predicted Train")
caret::confusionMatrix(conf_naive_train)
```

```
## Confusion Matrix and Statistics
##
##             Predicted Train
## Actual Train bad good
##         bad   62   52
##         good  55  231
##
##                Accuracy : 0.7325
##                  95% CI : (0.6863, 0.7753)
##     No Information Rate : 0.7075
##     P-Value [Acc > NIR] : 0.1480
##
##                   Kappa : 0.3488
##  Mcnemar's Test P-Value : 0.8467
##
##             Sensitivity : 0.5299
##             Specificity : 0.8163
##          Pos Pred Value : 0.5439
##          Neg Pred Value : 0.8077
##              Prevalence : 0.2925
##          Detection Rate : 0.1550
##    Detection Prevalence : 0.2850
##       Balanced Accuracy : 0.6731
##
##        'Positive' Class : bad
##
```

```r
conf_naive_test <- table(test$good_bad, predict_naive_test)
names(dimnames(conf_naive_test)) <- c("Actual Test", "Predicted Test")
caret::confusionMatrix(conf_naive_test)
```

```
## Confusion Matrix and Statistics
##
##            Predicted Test
## Actual Test bad good
##        bad   50   42
##        good  45  163
##
##                Accuracy : 0.71
##                  95% CI : (0.6551, 0.7607)
##     No Information Rate : 0.6833
##     P-Value [Acc > NIR] : 0.1762
##
##                   Kappa : 0.3242
##  Mcnemar's Test P-Value : 0.8302
##
```

```
##             Sensitivity : 0.5263
##             Specificity : 0.7951
##          Pos Pred Value : 0.5435
##          Neg Pred Value : 0.7837
##              Prevalence : 0.3167
##          Detection Rate : 0.1667
##    Detection Prevalence : 0.3067
##       Balanced Accuracy : 0.6607
##
##        'Positive' Class : bad
##
```

Analysis:

For the train dataset using NaiveBayes method we get accuracy 70% or misclassification of 30%, here we also notice that the accuracy of class 'bad' is 64.63% while for class 'good' is 72.24%, thus the model is more balanced in predicting, thus its not very baised in predict one class over the other.

For the test dataset using NaiveBayes method we get accuracy 66.8% or misclassification of 33.2%, here we also notice that the accuracy of class 'bad' is 63.40% while for class 'good' is 68.30%, thus the model is more balanced in predicting even compared to train.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
if (!require("pacman")) install.packages("pacman")
pacman::p_load(xlsx, ggplot2, MASS, tidyr, dplyr, reshape2, gridExtra,
               tree, caret, e1071)

set.seed(12345)
options("jtools-digits" = 2, scipen = 999)
credit_data <- read.xlsx("creditscoring.xls", sheetName = "credit")
credit_data$good_bad <- as.factor(credit_data$good_bad)
set.seed(12345)

n=NROW(credit_data)
set.seed(12345)
id=sample(1:n, floor(n*0.4))
train=credit_data[id,]

id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.3))
valid=credit_data[id2,]

id3=setdiff(id1,id2)
test=credit_data[id3,]

# Create a decision tree model
credit_tree_deviance <- tree(good_bad~., data=train, split = c("deviance"))
credit_tree_gini <- tree(good_bad~., data=train, split = c("gini"))

# Visualize the decision tree with rpart.plot
```

```r
summary(credit_tree_deviance)
summary(credit_tree_gini)

# predicting on the test dataset to get the misclassification rate.
predict_tree_deviance <- predict(credit_tree_deviance, newdata = test, type = "class")
predict_tree_gini <- predict(credit_tree_gini, newdata = test, type = "class")

conf_tree_deviance <- table(test$good_bad, predict_tree_deviance)
names(dimnames(conf_tree_deviance)) <- c("Actual Test", "Predicted Test")
caret::confusionMatrix(conf_tree_deviance)

conf_tree_gini <- table(test$good_bad, predict_tree_gini)
names(dimnames(conf_tree_gini)) <- c("Actual Test", "Predicted Test")
caret::confusionMatrix(conf_tree_gini)
set.seed(12345)


credit_tree <- tree(good_bad~., data=train, split = c("deviance"))

credit_tree_purned_train <- prune.tree(credit_tree, method = c("deviance"))
credit_tree_purned_test <- prune.tree(credit_tree, newdata = valid ,method = c("deviance"))

result_train <- cbind(credit_tree_purned_train$size, credit_tree_purned_train$dev, "Train")
result_test <- cbind(credit_tree_purned_test$size, credit_tree_purned_test$dev, "Valid")

result <- as.data.frame(rbind(result_test, result_train))
colnames(result) <- c("Leaf", "Deviance", "Type")

result$Leaf <- as.numeric(as.character(result$Leaf))
result$Deviance <- as.numeric(as.character(result$Deviance))


ggplot(data = result, aes(x = Leaf, y = Deviance, colour = Type)) + geom_point()

#Fitting the Naive Bayes model
credit_naive_model = naiveBayes(good_bad ~., data=train)

credit_naive_model


#Prediction on the dataset
predict_naive_train = predict(credit_naive_model, newdata=train)
predict_naive_test = predict(credit_naive_model, newdata=test)


conf_naive_train <- table(train$good_bad, predict_naive_train)
names(dimnames(conf_naive_train)) <- c("Actual Train", "Predicted Train")
caret::confusionMatrix(conf_naive_train)


conf_naive_test <- table(test$good_bad, predict_naive_test)
names(dimnames(conf_naive_test)) <- c("Actual Test", "Predicted Test")
caret::confusionMatrix(conf_naive_test)
```