Tutorial-7-MY-OWN-CODE

Dawn Cheung

2024-05-01

Decision Trees

Customer churn is the loss of clients or customers. Banks, telephone service, companies, Internet service providers, pay TV companies and insurance firms often use customer churn analysis and customer churn rates as one of their key business metrics. This is because the cost of retaining an existing customer is far less than acquiring a new one. Companies from these sectors often have customer service branches which attempt to win back defecting clients, because recovered long-term customers can be worth much more to a company than newly recruited clients. In this problem , a wireless telecommunications company wants to predict whether a customer will churn (switch to a different company) in the next six months. With a reasonably accurate prediction of a person's churning, the sales and marketing groups can attempt to retain the customer by ordering various incentives. Variables of our concern are listed below

i. Age (years)

ii. Married (true/false) [CATAGORICAL]

iii. Duration as a customer (years)

iv. Churned contacts -Number of the customer's contacts that have churned (count)

v. Churned (true/false) -Whether the customer churned

a. Build a decision tree for predicting customer churn, using the feature variables Age, Married, Cust_years and Churned_contacts.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.2
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to be
come errors
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.3.2
```

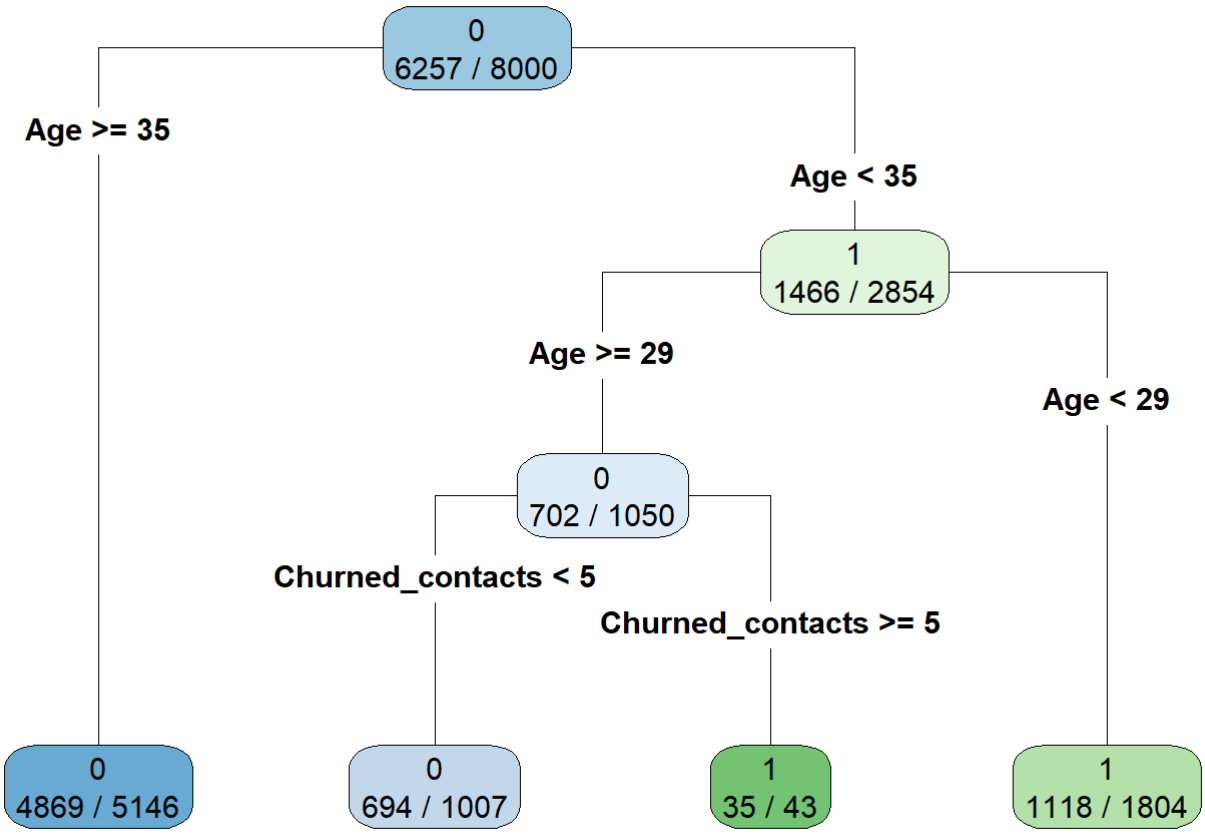
```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.2
```

```
data = read.csv("~/Github/DSA1101 Slayers/datasets/churn.csv")

data$Churned = as.factor(data$Churned)
data$Married = as.factor(data$Married)
glimpse(data)
```

```
## Rows: 8,000
## Columns: 6
## $ ID               <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16…
## $ Churned          <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,…
## $ Age              <int> 61, 50, 47, 50, 29, 43, 50, 29, 32, 48, 59, 35, 42, 3…
## $ Married          <fct> 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,…
## $ Cust_years       <int> 3, 3, 2, 3, 1, 4, 2, 3, 3, 4, 5, 5, 3, 3, 2, 2, 1, 3,…
## $ Churned_contacts <int> 1, 2, 0, 3, 3, 3, 1, 2, 3, 0, 2, 0, 1, 1, 2, 1, 2, 2,…
```

```
#maybe good habit to remove the ID column

attach(data)

fit <- rpart(Churned ~ Age + Married + Cust_years + Churned_contacts,
             method = "class",
             data = data,
             control = rpart.control(minsplit = 1),
             parms = list(split = "information"))

rpart.plot(fit, type = 4, extra = 2, clip.right.labs = FALSE)
```



b. Consider the decision tree in part (a) to predict binary variable Churned. Use the tree to predict customer churn for the following observations.

```
WHY = data.frame(Age = c(26, 23,56,36,45,28,22,22,60,32),
           Married = as.factor(c(1,1,1,1,0,0,1,0,1,0)),
           Cust_years = c(2,3,5,5,2,2,3,3,2,3),
           Churned_contacts = c(2,3,2,2,1,2,0,2,1,1))

predict(fit, newdata = WHY, type = "class")
```

```
##  1  2  3  4  5  6  7  8  9 10
##  1  1  0  0  0  1  1  1  0  0
## Levels: 0 1
```

2. (DT and N-fold Cross Validation) Consider the famous Iris Flower Data set which was first introduced in 1936 by the famous statistician Ronald Fisher. This data set consists of 50 observations from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each observation: the length and the width of the sepals and petals (in cm). In Tutorial 6, we used decision tree method to predict Iris species based on all four features. We now would want to use N-fold CV to check on how good the method is, based on the accuracy. We'll use 5-fold CV where we would want to keep the ratio of the three species the same (1:1:1) in both training set and test set.

What's the average accuracy of the decision tree method?

```
library(tidyverse)
library(rpart)
library(rpart.plot)
iris = read.csv("~/Github/DSA1101 Slayers/datasets/iris.csv")

glimpse(iris)
```

```
## Rows: 150
## Columns: 5
## $ sepal.length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.…
## $ sepal.width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.…
## $ petal.length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.…
## $ petal.width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.…
## $ class        <chr> "Iris-setosa", "Iris-setosa", "Iris-setosa", "Iris-setosa…
```

```
iris$class = as.factor(iris$class)
attach(iris) #is class supp to be a factor(??)

fit <- rpart(class ~ sepal.length + sepal.width + petal.length + petal.width,
             method = "class",
             data = iris,
             control = rpart.control(minsplit = 1),
             parms = list(split = "information"))

#OK OG IDEA WAS TOO COMPLICATED LMAO
#better idea: split setosa, virginica and vericolor
# then split each of those 3 grps into 5 groups, randomly
# then add 1 fold from each of the seperated sub-groups

n_folds=5 #

folds_j_1 <- sample(rep(1:n_folds, length.out = 50 ))  # for type 1 = setosa

folds_j_2 <- sample(rep(1:n_folds, length.out = 50 ))  # for type 2 = versicolor

folds_j_3 <- sample(rep(1:n_folds, length.out = 50 ))  # for type 2 = virginica


table(folds_j_1)
```

```
## folds_j_1
##  1  2  3  4  5
## 10 10 10 10 10
```

```
table(folds_j_2)
```

```
## folds_j_2
##  1  2  3  4  5
## 10 10 10 10 10
```

```
table(folds_j_3)
```

```
## folds_j_3
##  1  2  3  4  5
## 10 10 10 10 10
```

```
data1 = iris[1:50,] # data for type 1 = setosa
data2 = iris[51:100,] # data for type 2 = versicolor
data3 = iris[101:150,] # data for type 3 = virginica

acc=numeric(n_folds)

j= 1

for (j in 1:n_folds) {

test1 <- which(folds_j_1 == j)
test2 <- which(folds_j_2 == j)
test3 <- which(folds_j_3 == j)

train.1=data1[ -test1, ]
train.2=data2[ -test2, ]
train.3=data3[ -test3, ]

train = rbind(train.1, train.2, train.3) # this is the training data set

test = rbind(data1[test1,], data2[test2,], data3[test3,] ) # test data

fit.iris <- rpart(class ~ .,
method = "class", data =train, control = rpart.control( minsplit =1),
parms = list(split ='gini'))


pred = predict(fit.iris, newdata = test[,1:4], type = 'class')

confusion.matrix = table(pred, test[,5])

acc[j] = sum(diag(confusion.matrix))/sum(confusion.matrix)

}
acc
```

```
## [1] 0.9000000 0.9666667 0.8666667 0.8666667 1.0000000
```

```
mean(acc) # the accuracy is very high, 0.94
```

```
## [1] 0.92
```

3. Recall that we studied N-fold cross-validation for the K-nearest neighbor classifier, in which the value of k is varied to control the complexity of the decision surface for the classifier. For decision tree classification, when fitting a tree using function rpart(), we use the argument control = rpart.control(minsplit =1) where minsplit = 1 is to specify the minimum number of observations that must exist in a node in order for a split to be attempted. By default, minsplit = 20. This minsplit argument helps

to draft the complexity of a tree, complex with many layers and branches or simple with few layers and less branches.

For this control = rpart.control(), there is a similar complexity parameter exists, which is denoted as cp where by default cp = 0.01: control = rpart.control(cp = 0.01).

Heuristically, smaller values of cp correspond to decision trees of larger sizes, and hence more complex decision surfaces. For this problem, we will investigate n-fold cross validation for a decision tree classfier.

Consider the data set `bank-sample.csv' we discussed in the lectures. For this exercise, we will fit a decision tree with subscribed as outcome and job, marital, education, default, housing, loan, contact and poutcome as feature variables. We want to find the best cp value in terms of mis-classification error rate

a. Randomly split the entire data set into 10 mutually exclusive data sets.

```
banktrain = read.csv("~/Github/DSA1101 Slayers/datasets/bank-sample.csv", header = T)

drops = c("age", "balance", "day", "campaign", "pdays", "previous", "month", "duration")

banktrain = banktrain[,!names(banktrain) %in% drops]
# returns a list of boonleans

attach(banktrain)
head(banktrain)
```

```
##            job  marital education default housing loan  contact poutcome
## 1   management   single  tertiary      no     yes   no cellular  unknown
## 2 entrepreneur  married  tertiary      no     yes  yes cellular  unknown
## 3     services divorced secondary      no      no   no cellular  unknown
## 4   management  married  tertiary      no     yes   no cellular  unknown
## 5   management  married secondary      no     yes   no  unknown  unknown
## 6   management   single  tertiary      no     yes   no  unknown  unknown
##   subscribed
## 1         no
## 2         no
## 3        yes
## 4         no
## 5         no
## 6         no
```

```
head(banktrain)
```

```
##            job  marital education default housing loan  contact poutcome
## 1   management   single  tertiary      no     yes   no cellular  unknown
## 2 entrepreneur  married  tertiary      no     yes  yes cellular  unknown
## 3     services divorced secondary      no      no   no cellular  unknown
## 4   management  married  tertiary      no     yes   no cellular  unknown
## 5   management  married secondary      no     yes   no  unknown  unknown
## 6   management   single  tertiary      no     yes   no  unknown  unknown
##   subscribed
## 1         no
## 2         no
## 3        yes
## 4         no
## 5         no
## 6         no
```

```
## total records in dataset
n = dim(banktrain)[1]; n
```

```
## [1] 2000
```

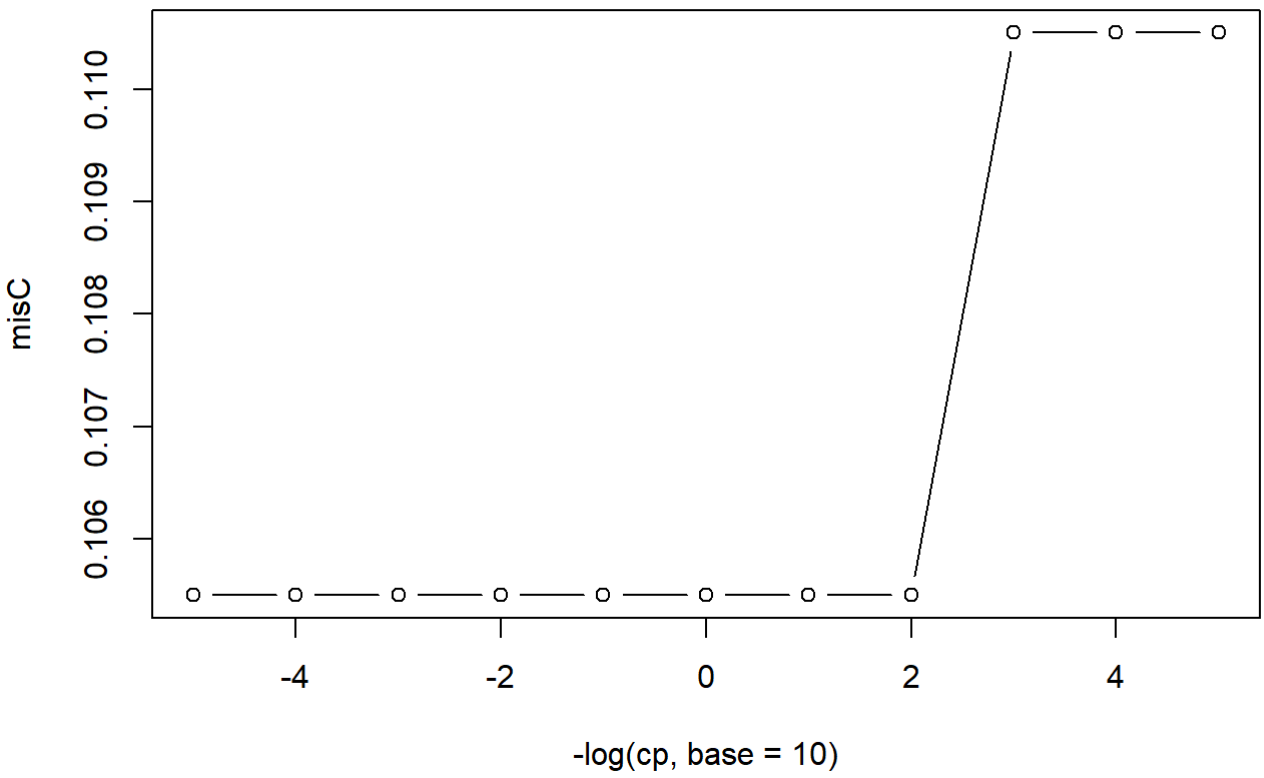
```
length(which(banktrain[,9] =="yes")) # 211 out of 2000 customers subscribed.
```

```
## [1] 211
```

```
## We'll randomly split data into 10 sets of (about) equal size
# regardless of percentage of "yes" in each set.

n_folds=10
folds_j <- sample(rep(1:n_folds, length.out = n))
# this is to randomly sample the indexes of subsets for the observation
#table(folds_j)

cp=10^(-5:5); length(cp)
```

```
## [1] 11
```

```
misC=rep(0,length(cp)) # a vector to record the rate of mis-classification for each cp

for(i in 1:length(cp)){
  misclass = 0
    for (j in 1:n_folds) {
        test = which(folds_j == j)
        train = banktrain[-c(test),]
        fit <- rpart(subscribed ~ job + marital +
                education + default + housing +
                loan + contact+poutcome,
                method="class",
                data=train,
                control=rpart.control(cp=cp[i]),
                parms=list(split='information'))
        new.data=data.frame(banktrain[test,c(1:8)])
    ##predict label for test data based on fitted tree
        pred = predict(fit,new.data,type='class')
        misclass = misclass + sum(pred != banktrain[test,9])
        }
    misC[i]=misclass/n
}

plot(-log(cp,base=10),misC,type='b')
```




```
#smth's very wrong here but uhhh yeah concept shld be there LOL
```

b. Let cp take on the values 10k for k = −5, −4, …, 0, …, 3, 4, 5.

```
#helloes
#cp =
```

c. At each cp value, run the following loop for j = 1, 2, …, 10:

i. Set the jth group to be the test set.
ii. Fit a decision tree on the other 9 sets with the value of cp.
iii. Predict the class assignment of subscribed for each observation of the test set.
iv. Calculate the number of mis-classification(s) by comparing predicted versus actual class labels in the test set.

```
#helloes
```

d. Determine the best cp value in terms of mis-classification error rate

```
print("huh")
```

```
## [1] "huh"
```