Assignment 2. Decision trees and logistic regression for bank marketing

Muhammad Umair

2022-12-04

Import the data to R, remove variable "duration" and divide into training/validation/test as 40/30/30

```
bank_data<-read.csv("bank-full.csv", sep = ";")

bank_data[,17] <- as.factor(bank_data[,17])
bank_data[,1] <- as.numeric(bank_data[,1])
bank_data[,2] <- as.factor(bank_data[,2])
bank_data[,3] <- as.factor(bank_data[,3])
bank_data[,4] <- as.factor(bank_data[,4])
bank_data[,5] <- as.factor(bank_data[,5])
bank_data[,6] <- as.numeric(bank_data[,6])
bank_data[,7] <- as.factor(bank_data[,7])
bank_data[,8] <- as.factor(bank_data[,8])
bank_data[,9] <- as.factor(bank_data[,9])
bank_data[,10] <- as.factor(bank_data[,10])
bank_data[,11] <- as.factor(bank_data[,11])
bank_data[,12] <- as.numeric(bank_data[,12])
bank_data[,13] <- as.numeric(bank_data[,13])
bank_data[,14] <- as.numeric(bank_data[,14])
bank_data[,15] <- as.numeric(bank_data[,15])
bank_data[,16] <- as.factor(bank_data[,16])
bank_data<-bank_data[,-12]

n<-dim(bank_data)[1]
set.seed(12345)
id<-sample(1:n, floor(n*0.4))
train<-bank_data[id,]
id1<-setdiff(1:n, id)
set.seed(12345)
id2<-sample(id1, floor(n*0.3))
valid<-bank_data[id2,]
id3<-setdiff(id1,id2)
test<-bank_data[id3,]
```

Fit decision trees to the training data so that you change the default settings one by one (i.e. not simultaneously):

a. Decision Tree with default settings.

```
require(tree)
```

```
## Loading required package: tree
```

```
## Warning: package 'tree' was built under R version 4.2.2
```

```
model_train1<-tree(y~ .,data = train)
predict_train1 <- predict(model_train1, newdata = train, type = "class")
```

b. Decision Tree with smallest allowed node size equal to 7000.

```
model_train2 <- tree(y ~ ., data = train, control = tree.control(nobs = nrow(train), minsize = 7000))
predict_train2 <- predict(model_train2, newdata = train, type = "class")
```

c. Decision trees minimum deviance to 0.0005.

```
model_train3 <- tree(y ~ ., data = train, control = tree.control(nobs = nrow(train), mindev = 0.0005))
predict_train3 <- predict(model_train3, newdata = train, type = "class")
```

Validation data

```
predict_Valid1 <- predict(model_train1, newdata = valid, type = "class")
predict_Valid2 <- predict(model_train2, newdata = valid, type = "class")
predict_Valid3 <- predict(model_train3, newdata = valid, type = "class")
```

#Confusion matrix of training data

```
c_matrix_train1 <- table(train$y, predict_train1)
c_matrix_train2 <- table(train$y, predict_train2)
c_matrix_train3 <- table(train$y, predict_train3)
```

#Confusion matrix of validation data

```
c_matrix_valid1 <- table(valid$y, predict_Valid1)
c_matrix_valid2 <- table(valid$y, predict_Valid2)
c_matrix_valid3 <- table(valid$y, predict_Valid3)
```

#MisClassification error of training data

```
Class_error_train1 <- (sum(c_matrix_train1) - sum(diag(c_matrix_train1))) / sum(c_matrix_train1)
Class_error_train2 <- (sum(c_matrix_train2) - sum(diag(c_matrix_train2))) / sum(c_matrix_train2)
Class_error_train3 <- (sum(c_matrix_train3) - sum(diag(c_matrix_train3))) / sum(c_matrix_train3)
```

#MisClassification error of Validation data

```
Class_error_valid1 <- (sum(c_matrix_valid1) - sum(diag(c_matrix_valid1))) / sum(c_matrix_valid1)
Class_error_valid2 <- (sum(c_matrix_valid2) - sum(diag(c_matrix_valid2))) / sum(c_matrix_valid2)
Class_error_valid3 <- (sum(c_matrix_valid3) - sum(diag(c_matrix_valid3))) / sum(c_matrix_valid3)
```

MisClassification error resuts

```
C_error_mat_train<-matrix(ncol = 3)
C_error_mat_valid<-matrix(ncol = 3)

C_error_mat_train<-rbind(Class_error_train1,Class_error_train2,Class_error_train3)
C_error_mat_valid<-rbind(Class_error_valid1,Class_error_valid2,Class_error_valid3)
print(C_error_mat_train)
```

```
##                          [,1]
## Class_error_train1 0.10484406
## Class_error_train2 0.10484406
## Class_error_train3 0.09245742
```

```
print(C_error_mat_valid)
```

```
##                         [,1]
## Class_error_valid1 0.1092679
## Class_error_valid2 0.1092679
## Class_error_valid3 0.1151663
```

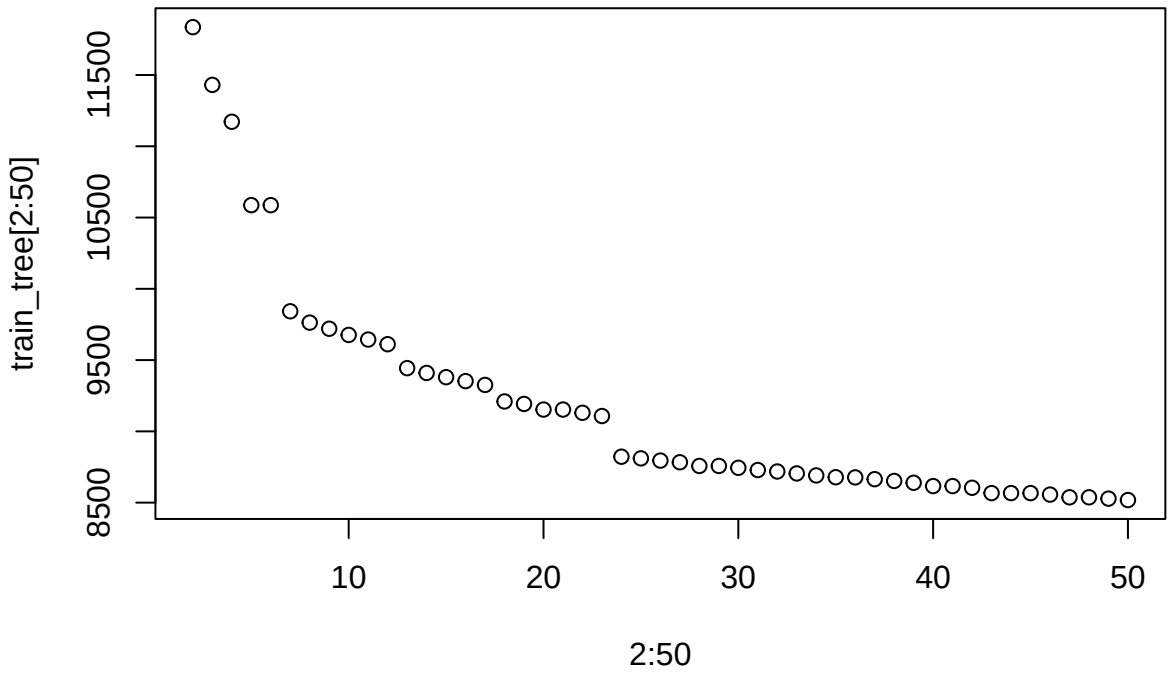
As it can be analyzed from the above results that Misclassification error of decision tree model with default settings and with node size 7000 is same for training and validation data set and Misclassification error of decision tree model with minimum deviance to 0.0005 is higher on validation data. In first decision tree model with default settings the smallest node size is 10 and in second model the smallest node size is 7000, so second model is best according to our dataset because increasing the node size decrease the tree size and it fits the model in better way. In default setting minimum deviance is 0.01 and in Third decision tree model we choosed minimum deviance to 0.0005 which increase the tree size and cause higher overfitting problem and we can see in misclassification error on model three.

Optimal tree depth in the model 2c: study the trees up to 50 leaves

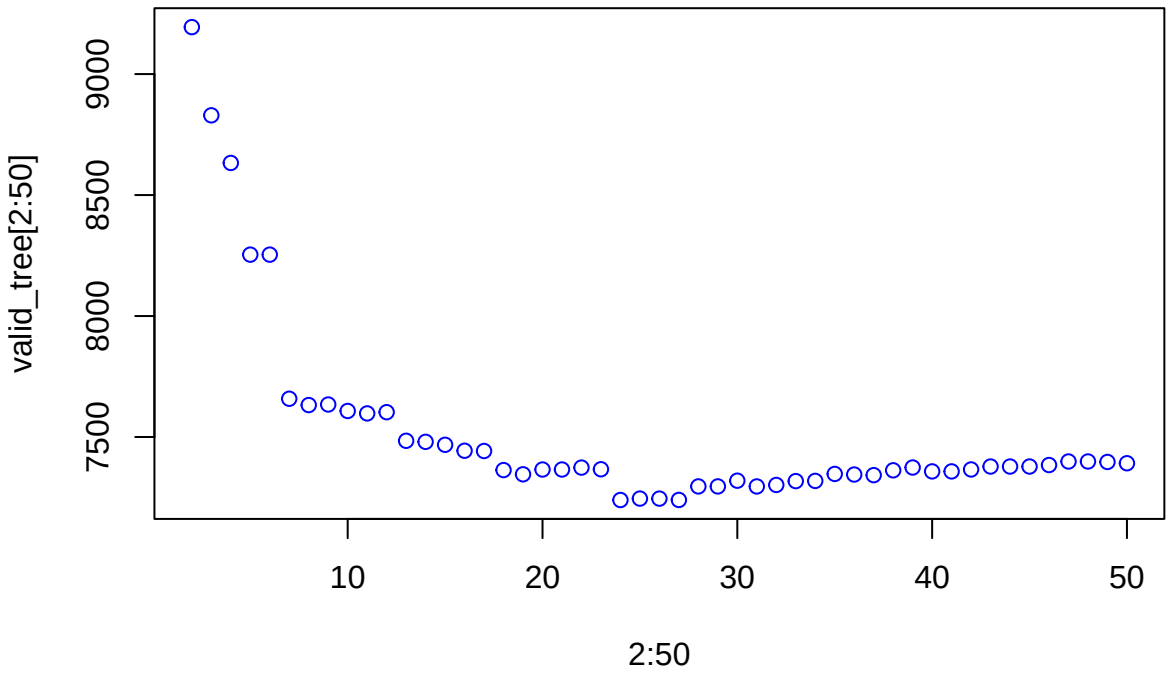
```
train_tree <- list()
valid_tree <- list()
train_tree[1]<-Inf
valid_tree[1]<-Inf
for(level in 2:50) {
  Tree_levels <- prune.tree(model_train3, best = level,method = "deviance")
  pred_train <- predict(Tree_levels, newdata = train, type = "tree")

  pred_valid <- predict(Tree_levels, newdata = valid, type = "tree")
  train_tree[level] <- deviance(pred_train)
  valid_tree[level] <- deviance(pred_valid)
}
opt_mat<-matrix(ncol = 2,nrow = 2)
colnames(opt_mat)<-c("Minimum deviance","Optimal Value")
opt_mat[1,]<-c(min(unlist(train_tree)),which.min(unlist(train_tree)))
opt_mat[2,]<-c(min(unlist(valid_tree)),which.min(unlist(valid_tree)))
print(opt_mat)
```

```
##      Minimum deviance Optimal Value
## [1,]         8518.088            50
## [2,]         7239.649            24
```

```
plot(2:50, train_tree[2:50], type="p", col="black")
```



```
plot(2:50, valid_tree[2:50], type="p", col="blue")
```




```
test_Tree_levels <- prune.tree(model_train3, best = 24,method = "deviance")
summary(test_Tree_levels)
```

```
##
## Classification tree:
## snip.tree(tree = model_train3, nodes = c(154L, 288L, 599L, 305L,
## 79L, 32L, 11L, 75L, 298L, 6L, 304L, 33L, 10L, 7L, 153L, 145L,
## 598L, 78L, 148L, 289L))
## Variables actually used in tree construction:
## [1] "poutcome" "month"    "contact"  "day"      "age"      "pdays"    "job"
## [8] "campaign" "housing"
## Number of terminal nodes:  24
## Residual mean deviance:  0.5633 = 10170 / 18060
## Misclassification error rate: 0.1027 = 1857 / 18084
```

As from the graph it can be seen that the optimal depth of tree on validation data is 24 and summary shows that 9 elements are used in tree classification that are more significant.

Confusion matrix, Acuuracy, F1 Score on test data

```
pred_test <- predict(test_Tree_levels, newdata = test, type = "class")
c_matrix_test <- table(test$y, pred_test)
print(c_matrix_test)
```

```
##      pred_test
##          no   yes
##   no  11783   196
##   yes  1255   330
```

```
Accuracy_test <- (sum(diag(c_matrix_test)) / sum(c_matrix_test))*100
cat("Accuracy on test data is =", Accuracy_test,"%")
```

```
## Accuracy on test data is = 89.30257 %
```

```
percision<-(c_matrix_test[1,1]/(c_matrix_test[1,1]+c_matrix_test[2,1]))
recal<-(c_matrix_test[1,1]/(c_matrix_test[1,1]+c_matrix_test[1,2]))
F1_score<-2*((percision*recal)/(percision+recal))
cat("F1_score on test data is =", F1_score)
```

```
## F1_score on test data is = 0.9419994
```

On test data the accuracy is 89% and F1 score is 0.94 so in this case we will prefer F1 score. And we can can say that our model has good predictive power. As we have imbalance dataset so F1 is balancing precision and recall so we will prefer F1.

Decision tree with loss function

```
require(rpart)
```

```
## Loading required package: rpart
```

```
## Warning: package 'rpart' was built under R version 4.2.2
```

```
require(caret)
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 4.2.2
```

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

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

```
loss_matr <- matrix(c(0, 1, 5, 0), nrow = 2, byrow = TRUE)

fit_loss <- rpart(y ~ ., data = train, method = "class", parms = list(loss = loss_matr))
pred_loss <- predict(fit_loss, newdata = test, type = "class")
c_matrix_loss <- table(test$y, pred_loss)
print(c_matrix_loss)
```

```
##      pred_loss
##          no   yes
##   no  10910  1069
##   yes   836   749
```

```
Accuracy_loss <- (sum(diag(c_matrix_loss)) / sum(c_matrix_loss))*100
cat("Accuracy on test data with loss matrix is =", Accuracy_loss,"%")
```

```
## Accuracy on test data with loss matrix is = 85.95547 %
```

After adding loss matrix the accuracy decreases because it reduce the chances 5 times to generate a false positive than a false negative. This will have the effect of making Class "no" predicted less frequently.

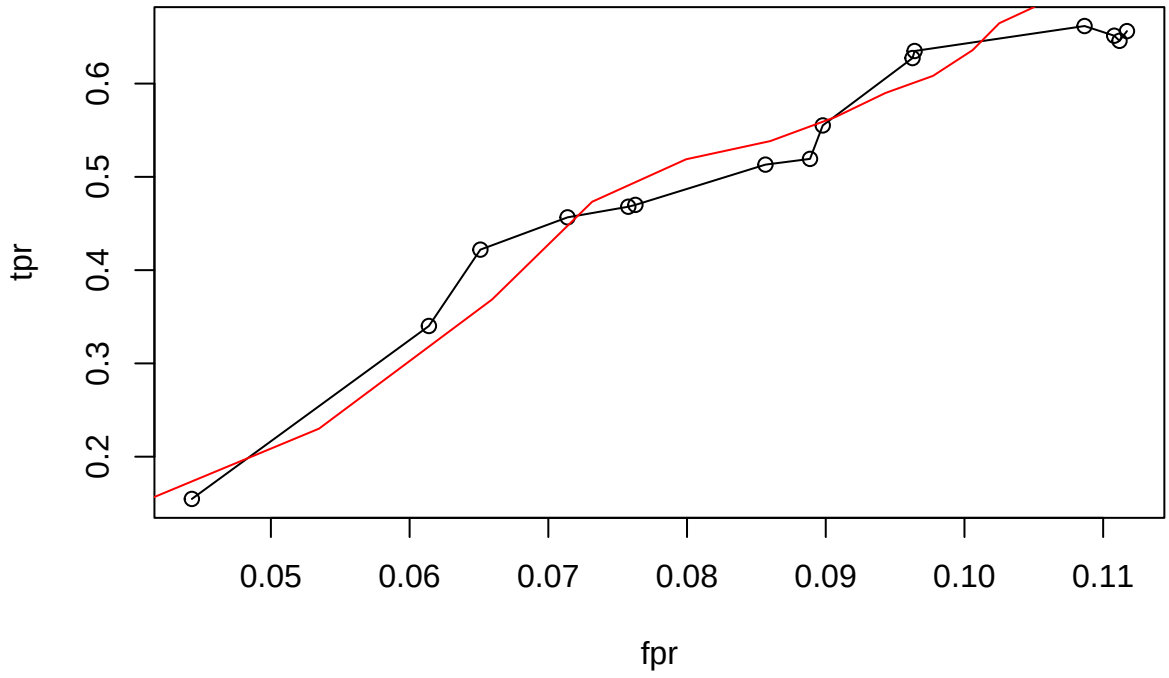
optimal tree and a logistic regression model

```
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 4.2.2
```

```
library(ggplot2)
p<-seq(0.05,0.95,by=0.05)
test_opt <- prune.tree(model_train3, best = 24,method = "deviance")
tpr_dc<-list()
fpr_dc<-list()
dc_df<-data.frame()
for(i in 1:15)
{
  pred_opt <- predict(test_opt, newdata = test, type = "vector")
  predict_thresh <- ifelse(pred_opt[,2] >p[i], "yes", "no")
  tb_dc<-table(predict_thresh,test$y)
  tpr_dc[i]<-tb_dc[2,2]/(tb_dc[2,1]+tb_dc[2,2])
  fpr_dc[i]<-tb_dc[1,2]/(tb_dc[1,1]+tb_dc[1,2])
}
dc_df = data.frame(fpr=unlist(fpr_dc),tpr=unlist(tpr_dc))
plot(dc_df,col="black")
lines(dc_df,col="black")
logistic_model <- glm(y ~ .,data = train,family = "binomial")
fpr_l<-list()
tpr_l<-list()
log_df<-data.frame()
for(i in 1:length(p))
```

```
{
  predict_reg <- predict(logistic_model, test , type = "response")
  predict_reg <- ifelse(predict_reg >p[i], "yes", "no")
  tb<-table(predict_reg,test$y)
  tpr_l[i]<-tb[2,2]/(tb[2,1]+tb[2,2])
  fpr_l[i]<-tb[1,2]/(tb[1,1]+tb[1,2])
}

log_df = data.frame(fpr=unlist(fpr_l),tpr=unlist(tpr_l))
lines(log_df,col="red")
```



ROC curve shows the tradeoff between true positive rate and false positive rate. ROC curve is better if dataset is balanced and in our case dataset is imbalanced between each class so percision recall curve is better. in this graph red line shows the logistic regression model and black line shows tha decision tree model.