lab2_block1

Shwetha

11/18/2020

Assignment 2

1

Partitioning the data into train , test and validation.

```
bank = read.csv("C:/Users/vcshw/Machine Learning and Stats/Sem1/Machine learning/lab/bank-full.csv", hea

bank = bank[,-12]
n = nrow(bank)
set.seed(12345)
id1=sample(1:n, floor(n*0.4))
train=bank[id1,]
d2 = bank[-id1,]
n2 = nrow(d2)
id2=sample(1:n2, floor(n2*0.5))
test=d2[id2,]
validate=d2[-id2,]
```

2

Fitting decision tree to training data

```
library(tree)
dt_default = tree(y~.,data = train)
dt_size = tree(y~.,data = train, control = tree.control(nrow(train),minsize = 7000))
dt_dev = tree(y~.,data = train, control = tree.control(nrow(train),mindev = 0.0005))
summary(dt_default)
```

```
##
## Classification tree:
## tree(formula = y ~ ., data = train)
## Variables actually used in tree construction:
## [1] "poutcome" "month"    "contact"  "housing"
## Number of terminal nodes:  6
## Residual mean deviance:  0.6022 = 10890 / 18080
## Misclassification error rate: 0.1048 = 1896 / 18084
```

```r
summary(dt_size)
```

```
##
## Classification tree:
## tree(formula = y ~ ., data = train, control = tree.control(nrow(train),
##     minsize = 7000))
## Variables actually used in tree construction:
## [1] "poutcome" "month"    "contact"
## Number of terminal nodes:  5
## Residual mean deviance:  0.6097 = 11020 / 18080
## Misclassification error rate: 0.1048 = 1896 / 18084
```

```r
summary(dt_dev)
```

```
##
## Classification tree:
## tree(formula = y ~ ., data = train, control = tree.control(nrow(train),
##     mindev = 5e-04))
## Variables actually used in tree construction:
##  [1] "poutcome"  "month"     "contact"   "marital"   "day"       "campaign"
##  [7] "job"       "pdays"     "age"       "balance"   "housing"   "education"
## [13] "previous"
## Number of terminal nodes:  122
## Residual mean deviance:  0.5213 = 9363 / 17960
## Misclassification error rate: 0.09362 = 1693 / 18084
```

```r
mce_default_train = summary(dt_default)$misclass[1]/summary(dt_default)$misclass[2]
mce_size_train = summary(dt_size)$misclass[1]/summary(dt_size)$misclass[2]
mce_dev_train = summary(dt_dev)$misclass[1]/summary(dt_dev)$misclass[2]

mce_default_train
```

```
## [1] 0.1048441
```

```r
mce_size_train
```

```
## [1] 0.1048441
```

```r
mce_dev_train
```

```
## [1] 0.09361867
```

For validation data

```r
y_default = predict(dt_default, validate, type = "class")
y_size = predict(dt_size, validate, type = "class")
y_dev = predict(dt_dev, validate, type = "class")

missclass = function(y,y1){
  n = length(y)
  return( 1 - sum(diag(table(y,y1)))/n)
}

mce_default_val = missclass(validate$y,y_default)
mce_size_val = missclass(validate$y,y_size)
mce_dev_val =  missclass(validate$y,y_dev)
mce_default_val
```

```
## [1] 0.1116927
```

```r
mce_size_val
```

```
## [1] 0.1116927
```

```r
mce_dev_val
```

```
## [1] 0.112946
```

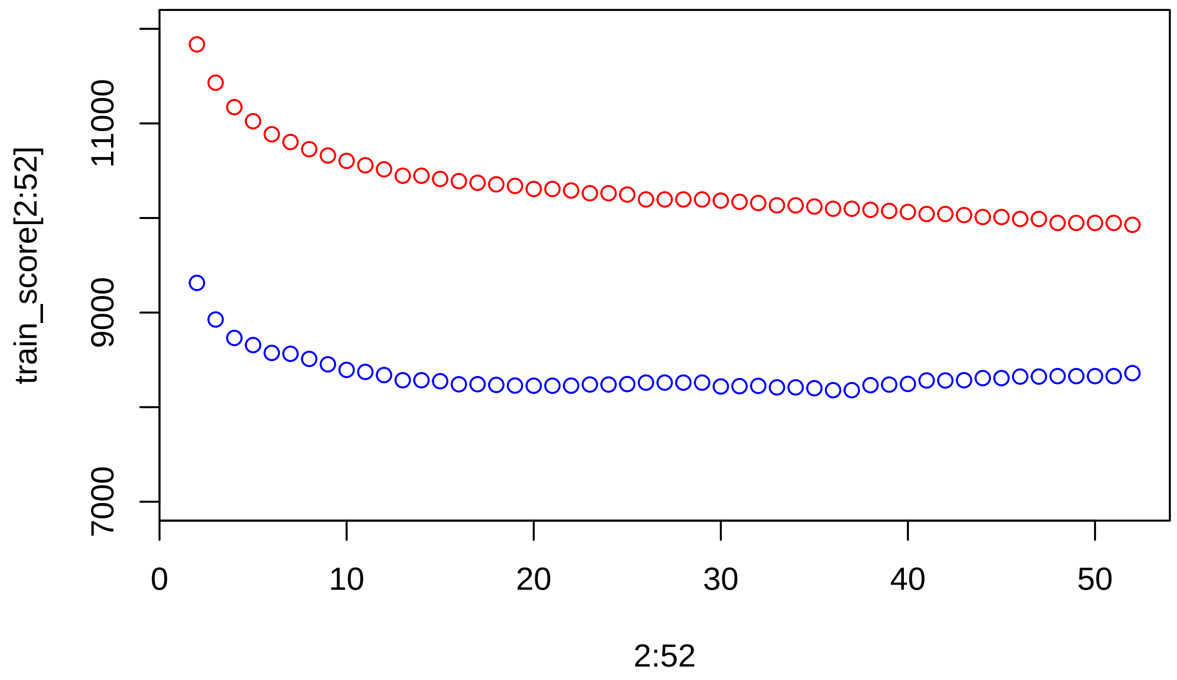
```r
# sensitive on data
```

report the misclassification rates for the training and validation data. Which model is the best one among these three? Report how changing the deviance and node size affected the size of the trees and explain why.

3

Selecting optimal tree by training and validation

```r
train_score = validate_score = rep(0,52)
for(i in 2:52){
  pruned_tree = prune.tree(dt_dev,best = i)
  pred = predict(pruned_tree, newdata=validate, type="tree")
  train_score[i] = deviance(pruned_tree)
  validate_score[i] = deviance(pred)
}
plot(2:52, train_score[2:52], type="p", col="red", ylim =c(7000,12000))
points(2:52, validate_score[2:52], type="p", col="blue")
```



Optimal amount of leaves = 24

«< which variables seem to be most important for decision making in this tree. Interpret the information provided by the tree structure (not everything but most important findings).

Confusion matrix and missclassification rate for test data.

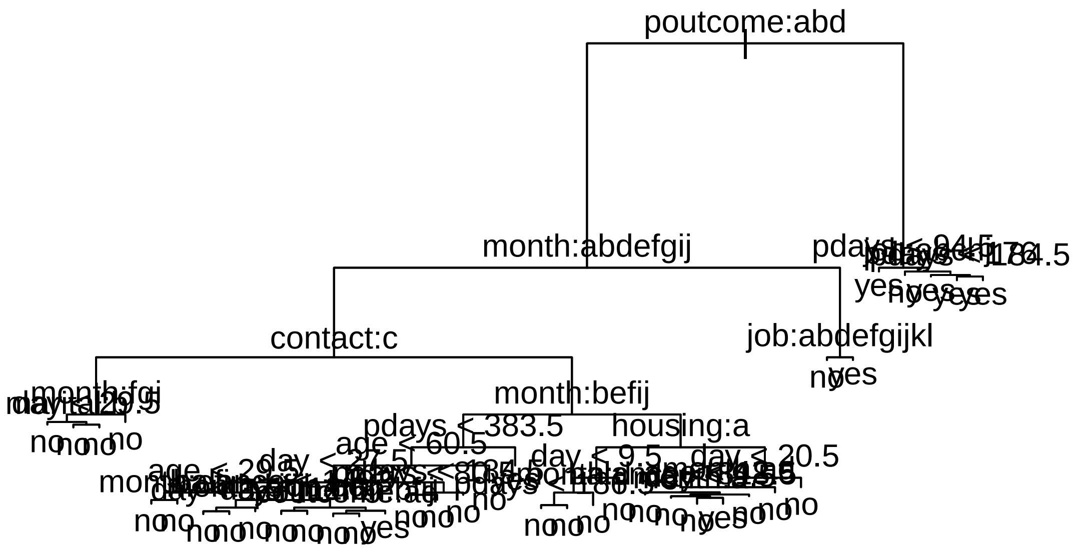
```
optimal_tree = prune.tree(dt_dev, best = 36)
optimal_pred = predict(optimal_tree, test, type = "class")
confusion_matrix = table(test$y,optimal_pred)
confusion_matrix
```

```
##      optimal_pred
##          no   yes
##   no  11868   122
##   yes  1347   226
```

```
missclassification_rate = missclass(test$y,optimal_pred)
missclassification_rate
```

```
## [1] 0.1083094
```

```
plot(optimal_tree)
text(optimal_tree)
```



```
summary(optimal_tree)
```

```
##
## Classification tree:
## snip.tree(tree = dt_dev, nodes = c(581L, 17L, 577L, 79L, 11L,
## 2506L, 37L, 66L, 77L, 304L, 30L, 14L, 312L, 576L, 153L, 580L,
## 2313L, 6L, 10L, 62L, 1157L))
## Variables actually used in tree construction:
##  [1] "poutcome" "month"    "contact"  "marital"  "day"      "pdays"
##  [7] "age"      "balance"  "job"      "housing"
## Number of terminal nodes:  37
## Residual mean deviance:  0.5595 = 10100 / 18050
## Misclassification error rate: 0.1026 = 1856 / 18084
```

4

```
loss_mat = t(matrix(c(0,1,5,0),2,2))
row.names(loss_mat) = colnames(loss_mat) = c("no","yes")
op = predict(optimal_tree, test)
loss_fit = ifelse(loss_mat[1,2]*op[,1] > loss_mat[2,1]*op[,2] ,"no","yes")#1 is no , 2 is yes
loss_confusion_matrix = table(test$y,loss_fit)

loss_mat
```

```
##     no yes
## no   0   1
## yes  5   0
```

```
loss_confusion_matrix
```

```
##      loss_fit
##          no   yes
##   no  10965  1025
##   yes   745   828
```

```
missclass(test$y,loss_fit)
```

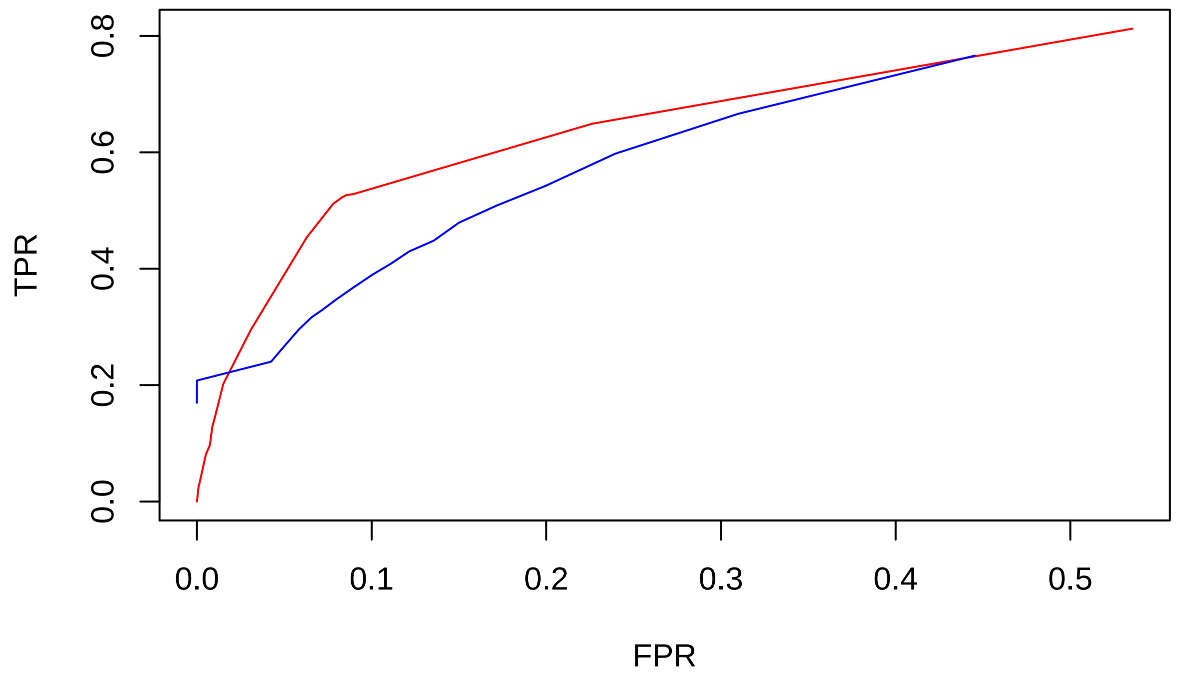
```
## [1] 0.1305021
```

Here in the loss function we can see that , penalty for predicting observed yes as no is 5 and no as yes is 1. So on applying loss function , as expected , the missclassification of an observed yes as no is reduced in the confusion matrix here. Previously observed yes predicted as no was 1227 , and now it is 780. However the missclassification error rate has increased.

5

Fitting naive bayes model

```
library(e1071)
naive_model = naiveBayes(y~., train)
naive_y = predict(naive_model, test, type = "raw")

pi = seq(0.05,0.95,0.05)
prob_naive_yes = naive_y[,2]
prob_dt_yes = op[,2]
real_y = ifelse(test$y == "yes",1,0) # yes is 1 , no is 0
NAIVE = DT = matrix(0,ncol = 3,nrow = length(pi))


for(i in 1:length(pi)){
  dt_assign = ifelse(prob_dt_yes > pi[i],1,0)
  naive_assign = ifelse(prob_naive_yes > pi[i],1,0)

  cm_dt = table(real_y,dt_assign)
  cm_naive = table(real_y,naive_assign)

  if(all(dim(cm_dt) == c(2,2))== TRUE){
  tpr_dt = cm_dt[2,2]/sum(cm_dt[2,])
  } else {
  tpr_dt = 0
  }
  if(all(dim(cm_naive) == c(2,2))){
  tpr_naive = cm_naive[2,2]/sum(cm_naive[2,])
  } else {
  tpr_naive = 0
  }

  if(all(dim(cm_dt) == c(2,2))== TRUE){
  fpr_dt = cm_dt[1,2]/sum(cm_dt[1,])
  }else {
  fpr_dt = 0
  }
  if(all(dim(cm_dt) == c(2,2))== TRUE){
  fpr_naive = cm_naive[1,2]/sum(cm_naive[1,])
  }else {
  fpr_naive = 0
  }

  NAIVE[i,] = c(pi[i],tpr_naive,fpr_naive)
  DT[i,] = c(pi[i],tpr_dt,fpr_dt)


}
colnames(NAIVE) = c("pi","TPR","FPR")
colnames(DT) = c("pi","TPR","FPR")

#ROC curve
plot(DT[,3],DT[,2],ylab = "TPR",xlab = "FPR", type = "l", col = "red")
lines(NAIVE[,3],NAIVE[,2],col= "blue")
```


TPR
FPR