MATH80619A Assignment 2 - Chapter 3 Report

Madi Kassymbekov

Question 1

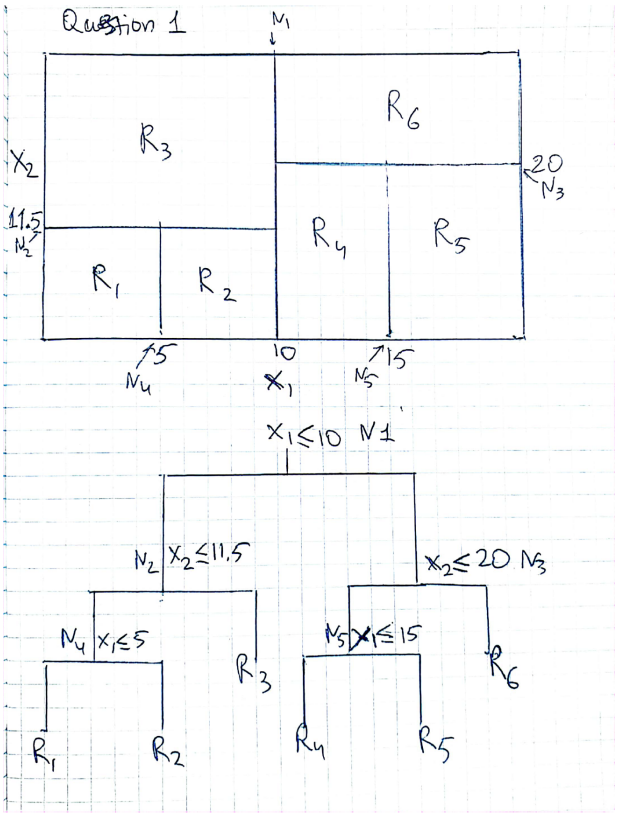

Figure 1: 6 Regions map and Decision Tree

Question 2

Majority vote technique

Most common class given the cutoff value will be selected which in this case is 1 as there are 6 probabilities above 0.5 and 4 under 0.5.

Average probability technique

Under this technique average probability from bootstrapped sample is calculated. In this case it is 0.45. Under the cutoff value of 0.5, the target predicted variable will be classified as 0.

Question 3

Compute the global error rate, false positive rate and false negative rate on the test set.

```
library(rpart)
library(randomForest)
library(party)
trainset <- read.csv("data/pimadiabetes_train.csv")
testset <- read.csv("data/pimadiabetes_test.csv")
set.seed(123456)

#Find optimal cut-off for binary classification
bestcutp=function(predcv,y,gainmat=diag(2),cutp=seq(0,1,.02),plotit=FALSE)
{
# predcv = vector of predicted probabilities (obtained out-of-sample by CV for example)
# y = vector of target (0-1)
# gainmat = gain matrix (2X2) (we want to maximize the gain)
# (1,1) = gain if pred=0 and true=0 (1,2) = gain if pred=0 and true=1
# (2,1) = gain if pred=1 and true=0 (2,2) = gain if pred=1 and true=1
# cutp=vector of thresholds to try
# plotit=plot or not the results
# value: a list with
# 1) matrix giving the thresholds and estimated mean gains
# 2) the threshold with maximum gain, with the associated mean gain
  nc=length(cutp) ; gain=rep(0,nc)
  for(i in 1:nc)
  {
  pred=as.numeric(predcv>cutp[i])
  gain[i]=mean(gainmat[1,1]*(pred==0)*(y==0)+gainmat[1,2]*(pred==0)*(y==1)+
  gainmat[2,1]*(pred==1)*(y==0)+gainmat[2,2]*(pred==1)*(y==1))
  }
  if(plotit){plot(cutp,gain,type="l",xlab="threshold",ylab="gain")}
  out=list(NULL,NULL)
  out[[1]]=cbind(cutp,gain)
  out[[2]]=out[[1]][which.max(gain),]
  out
}


#a) Grow a single tree using the CART algorithm based on a cutoff of 0.5 for prediction
cart.fit <- rpart(diabetes~., data=trainset, method="class")
cart.predict <- predict(cart.fit, newdata=testset, type="class")
confmatrix <- table(cart.predict, testset$diabetes)
print("CART confusion matrix")
```

```
## [1] "CART confusion matrix"
```

```
confmatrix
```

```
##
## cart.predict   0   1
##            0 107  29
##            1  23  41
```

```
error_rate <- mean(cart.predict!=testset$diabetes)
results <- data.frame("CART", mean(cart.predict!=testset$diabetes),
```

```
                    confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
                    confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
names(results) <- c("Model", "Error_Rate", "FalsePosRate", "FalseNegRate")

#b)Repeat question a) using an optimal cutoff minimizing the global error rate
#assuming an equal cost between false positive and false negative. In general,
#can you compare the error rates computed using different gain matrices
#(answer yes or no, and explain)?
predcartprob <- predict(cart.fit, newdata=testset, type="prob")
optcutoff <- bestcutp(predcartprob[,2], testset$diabetes, gainmat=diag(2),
cutp = seq(0,1,.02, plotit=FALSE))
optcutoff[[2]] #Optimal cut-off value
```

```
##  cutp  gain
## 0.200 0.755
```

```
predcartoptimal <- as.numeric(predcartprob[,2]>optcutoff[[2]][1])
confmatrix <- table(predcartoptimal, testset$diabetes)
print("CART optimal cut-off confusion matrix")
```

```
## [1] "CART optimal cut-off confusion matrix"
```

```
confmatrix
```

```
##
## predcartoptimal  0  1
##               0 97 16
##               1 33 54
```

```
results[nrow(results) + 1,] = list(Model="CART optimal cut-off",
                                   Error_Rate=mean(predcartoptimal!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
#c) Repeat questions a) and b) using a conditional tree
#Ordinary Conditional Tree
ct.fit <- ctree(as.factor(diabetes)~., data=trainset)
ct.predict <- predict(ct.fit, newdata=testset, type="response")
confmatrix <- table(ct.predict, testset$diabetes)
print("Conditional Tree confusion matrix")
```

```
## [1] "Conditional Tree confusion matrix"
```

```
confmatrix
```

```
##
## ct.predict   0   1
##          0 102  27
##          1  28  43
```

```
results[nrow(results) + 1,] = list(Model="Conditional Tree",
                                   Error_Rate=mean(ct.predict!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
#Optimal cutoff conditional tree
predctprob <- predict(ct.fit, newdata=testset, type="prob")
predctprob <- matrix(unlist(predctprob), ncol=2, byrow=TRUE)
optcutoff <- bestcutp(predctprob[,2], testset$diabetes, gainmat=diag(2),
cutp = seq(0,1,.02, plotit=FALSE))
```

```
optcutoff[[2]] #Optimal cut-off value
```

```
##  cutp  gain
## 0.600 0.735
```

```
predctoptimal <- as.numeric(predctprob[,2]>optcutoff[[2]][1])
confmatrix <- table(predctoptimal, testset$diabetes)
print("Conditional Tree optimal cut-off confusion matrix")
```

```
## [1] "Conditional Tree optimal cut-off confusion matrix"
```

```
confmatrix
```

```
##
## predctoptimal   0   1
##             0 122  45
##             1   8  25
```

```
results[nrow(results) + 1,] = list(Model="Conditional Tree optimal cut-off",
                                   Error_Rate=mean(predctoptimal!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
#d) Grow a random forest with B=500 trees using the CART algorithm
#and a cutoff of 0.5
rf.fit <- randomForest(as.factor(diabetes)~., data=trainset, ntree=500)
rf.predict<- predict(rf.fit, newdata=testset, type="response")
confmatrix <- table(rf.predict, testset$diabetes)
print("Random Forest CART based confusion matrix")
```

```
## [1] "Random Forest CART based confusion matrix"
```

```
confmatrix
```

```
##
## rf.predict   0   1
##          0 109  32
##          1  21  38
```

```
results[nrow(results) + 1,] = list(Model="Random Forest CART",
                                   Error_Rate=mean(rf.predict!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
#e) Grow a random forest with B=500 trees using the conditional tree algorithm
#and a cutoff of 0.5
cf.fit <- cforest(as.factor(diabetes)~., data=trainset,
controls = cforest_control(ntree = 500))
cf.predict<- predict(cf.fit, newdata=testset, type="response")
confmatrix <- table(cf.predict, testset$diabetes)
print("Random Forest Conditional Tree based confusion matrix")
```

```
## [1] "Random Forest Conditional Tree based confusion matrix"
```

```
confmatrix
```

```
##
## cf.predict   0   1
##          0 109  31
##          1  21  39
```

```
results[nrow(results) + 1,] = list(Model="Random Forest CTree",
                                   Error_Rate=mean(cf.predict!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
#f) Repeat questions d) and e) using the bagging approach
#CART Based RF Bagging
fulldata <- rbind(trainset, testset)
bag.rf.fit <- randomForest(as.factor(diabetes)~., data=fulldata,
subset = 1:568,
ntree=500)
bag.rf.predict <- predict(bag.rf.fit, newdata=testset)
confmatrix <- table(bag.rf.predict, testset$diabetes)
print("Random Forest CART Bagging based confusion matrix")
```

```
## [1] "Random Forest CART Bagging based confusion matrix"
```

```
confmatrix
```

```
##
## bag.rf.predict   0   1
##              0 111  34
##              1  19  36
```

```
results[nrow(results) + 1,] = list(Model="Random Forest CART Bagging",
                                   Error_Rate=mean(bag.rf.predict!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
#CTree Based RF Bagging
bag.cf.fit <- cforest(as.factor(diabetes)~., data=fulldata, subset = 1:568,
                          controls = cforest_control(ntree = 500))
bag.cf.predict <- predict(bag.cf.fit, newdata=testset)
confmatrix <- table(bag.cf.predict, testset$diabetes)
print("Random Forest CTree Bagging based confusion matrix")
```

```
## [1] "Random Forest CTree Bagging based confusion matrix"
```

```
confmatrix
```

```
##
## bag.cf.predict   0   1
##              0 109  33
##              1  21  37
```

```
results[nrow(results) + 1,] = list(Model="Random Forest CTree Bagging",
                                   Error_Rate=mean(bag.cf.predict!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
results
```

```
##                              Model Error_Rate FalsePosRate FalseNegRate
## 1                             CART      0.260   0.17692308    0.4142857
## 2             CART optimal cut-off      0.245   0.25384615    0.2285714
## 3                 Conditional Tree      0.275   0.21538462    0.3857143
## 4 Conditional Tree optimal cut-off      0.265   0.06153846    0.6428571
## 5               Random Forest CART      0.265   0.16153846    0.4571429
## 6              Random Forest CTree      0.260   0.16153846    0.4428571
## 7       Random Forest CART Bagging      0.265   0.14615385    0.4857143
```

```
## 8      Random Forest CTree Bagging      0.270   0.16153846    0.4714286
```

```
#g) Compare the results. Compute the variable importance measures based on the best model.
cart.fit$variable.importance
```

```
##  glucose     mass      age pressure  insulin pregnant pedigree  triceps
## 66.58684 27.30498 24.95488 19.61302 16.25810 13.47010 11.39796  8.54755
```

b, Can you compare the error rates computed using different gain matrices?

g. Compare the results.Compute the variable importance measures based on the best model.

Question 4 Multiple-Choice Questions

1. Which of the following statements is true?

A. Given the same dataset for both bagged decision tree and random forest, the number of predictors used at a single split will be equal between the two.

Answer: False. Random Forest uses a subset of predictors for model building, whereas bagged trees use all the features at once.

B. Generally on average, random forests perform better than bagged trees and single trees.

Answer: True. Random forests perform better in practice due to the fact that forest has less correlated trees on average compared to bagging trees. It's also true that ensemble method which is random forest on average perform better than single trees due to the weighted averaging of end prediction which will have better generalization performance.

C. Decision trees are affected by multicollinearity in features.

Answer: False. If, for example, 90% of features are correlated, decision tree will pick only one of them for splitting.

D. Random forest tree can be interpreted as well as CART.

Answer: False. Random forest is a black box and can't be used for interpretation due to the large number of trees and bigger depth than cart. Even if taking into account only one tree, it can be very deep with thousands of nodes which is unbearable for human to interpret and use as an explanatory model.

2. What can be concluded about Random forest tree algorithm?

A. Predictions from random forest generally have higher variance compared to decision trees.

Answer: False. Random forest have lower variance compared to decision tree due to attempts to reduce prediction variance by bootstrap aggregation i.e. training on different samples of the data.

B. Feature Normalization improves the prediction performance of random forests.

Answer: False. Normalizing features does not improve performance because each node in the random forest tree is simply splitting a sorted list and not comparing one feature's value to another feature's value..

C. Variable importance of random forest highlights significant features to be used in predicting target variable in general regardless of the model used.

Answer: False. Variable importance only showcases importance of features used in fitting a random forest model for the given training dataset. It should not be interpreted as a global significance of feature like in linear regression.

D. To reduce underfitting of random forest model, it is better to increase the depth of the tree.

Answer: True. Increasing depth will help random forest to learn from training data better as more features are used in splitting due to increase depth.