lab8

YILIN LI

2019/11/14

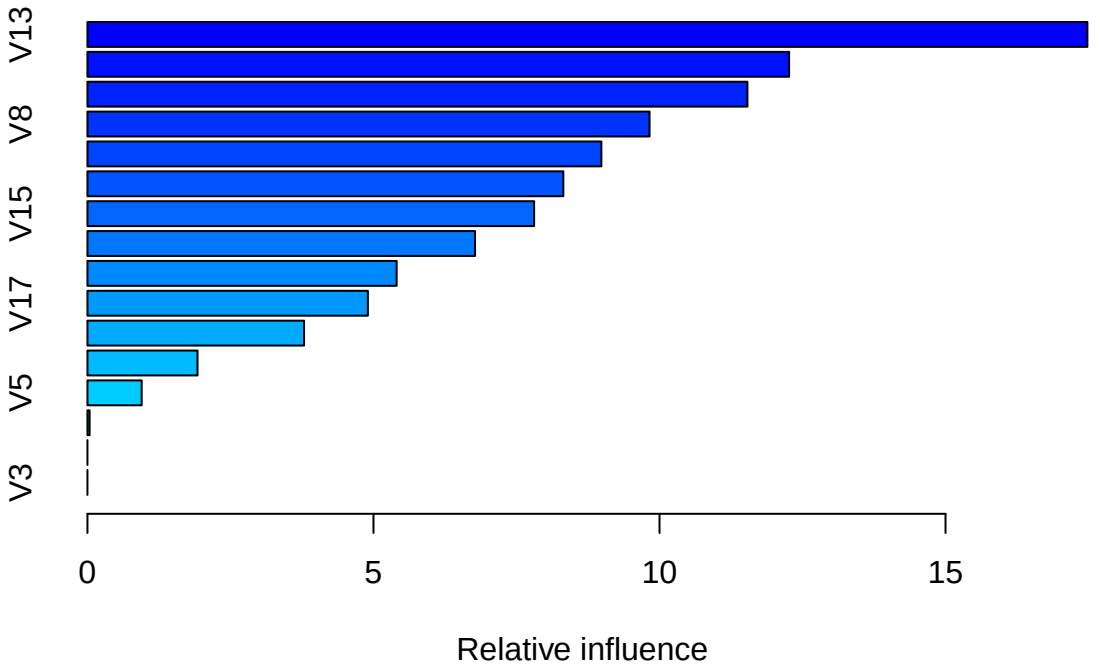
Lab 8

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
lettersdf <- read.csv("https://raw.githubusercontent.com/stat-learning/course-materials/master/data/let
                      header = FALSE)
set.seed(1)
train <- sample(1:nrow(lettersdf), nrow(lettersdf) * .75)
traindf <- lettersdf[train,]
testdf <- lettersdf[-train,]

bt1 <- gbm(V1~., data = traindf, distribution = "multinomial",interaction.depth = 1, shrinkage = 0.1, n
summary(bt1)
```




```
##     var     rel.inf
## V13 V13 17.47975897
## V12 V12 12.26575586
## V14 V14 11.53730240
## V8   V8  9.82580683
## V10 V10  8.98367860
## V11 V11  8.31923619
## V15 V15  7.80833554
## V9   V9  6.77590463
## V16 V16  5.40537511
```

```
## V17 V17  4.90210556
## V7   V7  3.78671390
## V4   V4  1.92421542
## V5   V5  0.94803360
## V6   V6  0.03777739
## V2   V2  0.00000000
## V3   V3  0.00000000
```

It turns out that "V13" is the most important predictor.

```
yhat <- predict(bt1, newdata = testdf, type = "response", n.trees = 50)
predicted <- LETTERS[apply(yhat, 1, which.max)]
tb <- table(predicted, testdf$V1)
MCR <- 1- sum(diag(tb))/length(predicted)
P <- diag(tb)/addmargins(tb)[-27,27]
R <- diag(tb)/addmargins(tb)[27,-27]
F1 <- 2*P*R/(P+R)
MCR
```

```
## [1] 0.3192
```

```
which.min(F1)
```

```
## E
## 5
```

1. Built as above
2. The misclassification rate is 0.3192
3. According to the lowest F1 score, letter "E" is the most difficult one to predict.
4. By scanning throught the confusion matrix, we can see that "B","D" and "Q","G" are two pairs of letters that difficult to distinguish each other.

```
bt2 <- gbm(V1~., data = traindf, distribution = "multinomial",interaction.depth = 1, shrinkage = 0.05, n
yhat2 <- predict(bt2, newdata = testdf, type = "response", n.trees = 500)
predicted2 <- LETTERS[apply(yhat2, 1, which.max)]
tb2 <- table(predicted2, testdf$V1)
MCR2 <- 1 - sum(diag(tb2))/length(predicted2)
MCR2
```

```
## [1] 0.1972
```

1. The misclassification rate of the new model is 0.1972 which is better than the previous one.
2. By scanning through the new confusion matrix, there is no obvious letter pairs that is hard to be distinguished from each other.

```
d <- read.csv("http://andrewpbray.github.io/data/crime-train.csv")
d["NumIllegsr"] <- sqrt(d$NumIlleg)
test_data <- read.csv("https://bit.ly/2PYS8Ap")
test_data["NumIllegsr"] = sqrt(test_data$NumIlleg)

bt3 <- gbm(ViolentCrimesPerPop~., data = d, distribution = "gaussian",interaction.depth = 1, shrinkage =
predicted3 <- predict(bt3, newdata = test_data, n.trees = 5000)
MSE_bt = mean((predicted3-test_data$ViolentCrimesPerPop)^2)
MSE_bt
```

```
## [1] 0.001967945
```

It turns out the test MSE for the boosted is 0.002 which is the lowest among all the other algorithms we learned so far, such as linear regression, decision tree, bagging tree, and random forest.

Chapter 8 Exercise

5.

```
p <- c(0.1,0.15,0.2,0.2,0.55,0.6,0.6,0.65,0.7,0.75)
majority <- sum(p > 0.5) >= length(p)/2
averprob <- mean(p)
majority
```

```
## [1] TRUE
```

```
averprob
```

```
## [1] 0.45
```

According to the first method, this result belongs to red class. The second method, however, classify this result in green class.

6. Firstly, set the threshold of the stopping condition, such as each terminals has lower than 5 observations. Then iterate through all the possible cut of each predictor and find the cut that has the lowest training MSE. Then split the node using that cut, resulting in a left node and a right node, which are subtrees of the original tree. Recursively use the above method to generate the full tree. Then apply the cost complexity pruning to the full tree and obtain best subtrees as a function of pruning parameter α. Then, use the K folds cross validation method, find average test runed error for each possible α. Choose the subtree with the best pruning parameter α as the final tree.