Course 8 Project

Haiyang Zhang

10/20/2019

Data Processing

1. Import the raw data and remove columns with many NA terms.

```
data.train<-read.csv("pml-training.csv")
data.test<-read.csv("pml-testing.csv")

NAcount<-nrow(data.train)/100*20
reCol<-which(colSums(is.na(data.train)|data.train=="")>NAcount)

classe_lvl<-levels(data.train$classe)
data.train$classe<-factor(data.train$classe, classe_lvl)

train<-data.frame(data.train[,-reCol])
train<-train[,-c(1:6)]
test<-data.frame(data.test[,-reCol])
test<-test[,-c(1:6)]
```

2. Split training set into a sub-training set and testing part.

```
set.seed(12596876)
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

classeInd <- which(names(train) == "classe")
partition <- createDataPartition(y=train$classe, p=0.75, list=FALSE)
subSetTrain <- train[partition, ]
subSetTest <- train[-partition, ]
```

3. Try to check if there are any terms correlated to the variable classe

```
correlations <- cor(subSetTrain[, -classeInd],
as.numeric(subSetTrain$classe))
bestCorr <- subset(as.data.frame(as.table(correlations)), abs(Freq)>0.3)
bestCorr

##             Var1 Var2      Freq
## 42 pitch_forearm    A 0.3449749
```

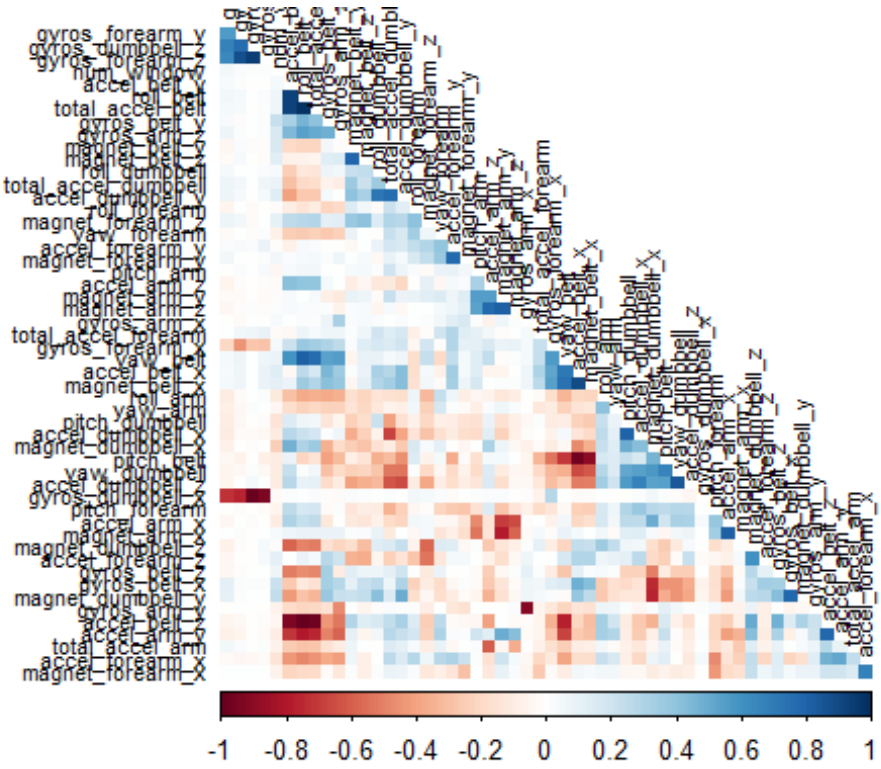
There is no strong correlation between classe with other variables.

Modeling

```
library(corrplot)

## corrplot 0.84 loaded

correlationMx <- cor(subSetTrain[, -classeInd])
highlyCorr <- findCorrelation(correlationMx, cutoff=0.9, exact=TRUE)
exColumns <- c(highlyCorr, classeInd)
corrplot(correlationMx, method="color", type="lower", order="hclust",
tl.cex=0.70, tl.col="black", diag = FALSE)
```



Random Forest by using 200 trees:

```
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

ntree <- 200
start <- proc.time()
```

```
rf.clean <- randomForest(
  x=subSetTrain[, -classeInd],
  y=subSetTrain$classe,
  xtest=subSetTest[, -classeInd],
  ytest=subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE)
proc.time() - start

##    user  system elapsed
##  119.42    1.30  123.55

start <- proc.time()
rf.exclude <- randomForest(
  x=subSetTrain[, -exColumns],
  y=subSetTrain$classe,
  xtest=subSetTest[, -exColumns],
  ytest=subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE)
proc.time() - start

##    user  system elapsed
##  114.64    2.06  119.88
```

Test

```
rf.clean

##
## Call:
##  randomForest(x = subSetTrain[, -classeInd], y = subSetTrain$classe,
xtest = subSetTest[, -classeInd], ytest = subSetTest$classe,      ntree =
ntree, proximity = TRUE, keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.26%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4184    0    0    0    1 0.0002389486
## B    5 2840    3    0    0 0.0028089888
## C    0    7 2560    0    0 0.0027269186
## D    0    0   17 2395    0 0.0070480929
## E    0    0    0    5 2701 0.0018477458
##                 Test set error rate: 0.33%
## Confusion matrix:
##      A   B   C   D   E class.error
```

```
## A 1395   0   0   0   0 0.000000000
## B    3 946   0   0   0 0.003161222
## C    0   5 850   0   0 0.005847953
## D    0   0   6 797   1 0.008706468
## E    0   0   0   1 900 0.001109878
```

```
rf.clean.acc <- round(1-sum(rf.clean$confusion[, 'class.error']),3)
paste0("Accuracy on training: ",rf.clean.acc)
```

```
## [1] "Accuracy on training: 0.985"
```

```
rf.clean.testing.acc <- round(1-sum(rf.clean$test$confusion[,
'class.error']),3)
paste0("Accuracy on testing: ",rf.clean.testing.acc)
```

```
## [1] "Accuracy on testing: 0.981"
```

```
rf.exclude
```

```
##
## Call:
##  randomForest(x = subSetTrain[, -exColumns], y = subSetTrain$classe,
xtest = subSetTest[, -exColumns], ytest = subSetTest$classe,      ntree =
ntree, proximity = TRUE, keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 0.24%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4185    0    0    0    0 0.0000000000
## B    2 2841    5    0    0 0.0024578652
## C    0    8 2558    1    0 0.0035060382
## D    0    0   17 2395    0 0.0070480929
## E    0    0    0    2 2704 0.0007390983
##                 Test set error rate: 0.37%
## Confusion matrix:
##      A   B   C   D   E class.error
## A 1395   0   0   0   0 0.000000000
## B    3 946   0   0   0 0.003161222
## C    0   5 850   0   0 0.005847953
## D    0   0   7 796   1 0.009950249
## E    0   0   0   2 899 0.002219756
```

```
rf.exclude.acc <- round(1-sum(rf.exclude$confusion[, 'class.error']),3)
paste0("Accuracy on training: ",rf.exclude.acc)
```

```
## [1] "Accuracy on training: 0.986"
```

```
rf.exclude.testing.acc <- round(1-sum(rf.exclude$test$confusion[,
'class.error']),3)
paste0("Accuracy on testing: ",rf.exclude.testing.acc)

## [1] "Accuracy on testing: 0.979"
```

pred<-t(cbind(clean=as.data.frame(predict(rf.clean, test),optional=T), exclude=as.data.frame(predict(rf.exclude,test[,-exColumns],optional=T)))) pred

Conclusion

The second model is better with lower error and better accuracy.