Prediction Assignment Writeup

Animesh Ranjan

10/20/2016

### Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement Â– a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks.

One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this data set, the participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: <http://groupware.les.inf.puc-rio.br/har> (see the section on the Weight Lifting Exercise Dataset).

In this project, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants toto predict the manner in which praticipants did the exercise.

The dependent variable or response is the “classe” variable in the training set.

## Loaded the data in R

The data are loaded in R

trainingOrg <- read.csv("~/Desktop/JHU Data Science/Practical Machine Learning/assignment/pml-training.csv", na.strings=c("", "NA", "NULL"))  
  
testingOrg <- read.csv("~/Desktop/JHU Data Science/Practical Machine Learning/assignment/pml-testing.csv", na.strings=c("", "NA", "NULL"))

dim(trainingOrg)

## [1] 19622 160

dim(testingOrg)

## [1] 20 160

## Data preparation

Removing variables having too many NA values.

training.dena <- trainingOrg[ , colSums(is.na(trainingOrg)) == 0]  
dim(training.dena)

## [1] 19622 60

Removing unrelevant variables which are unlikely to be related to dependent variable.

remove = c('X', 'user\_name', 'raw\_timestamp\_part\_1', 'raw\_timestamp\_part\_2', 'cvtd\_timestamp', 'new\_window', 'num\_window')  
training.dere <- training.dena[, -which(names(training.dena) %in% remove)]  
dim(training.dere)

## [1] 19622 53

Checking the variables that have extremely low variance

library(caret)  
  
zeroVar= nearZeroVar(training.dere[sapply(training.dere, is.numeric)], saveMetrics = TRUE)  
training.nonzerovar = training.dere[,zeroVar[, 'nzv']==0]  
dim(training.nonzerovar)

## [1] 19622 53

Remove highly correlated variables 90%

corrMatrix <- cor(na.omit(training.nonzerovar[sapply(training.nonzerovar, is.numeric)]))  
dim(corrMatrix)

## [1] 52 52

There are 52 variables left

corrDF <- expand.grid(row = 1:52, col = 1:52)  
corrDF$correlation <- as.vector(corrMatrix)

We are going to remove those variable which have high correlation.

removecor = findCorrelation(corrMatrix, cutoff = .90, verbose = FALSE)  
training.decor = training.nonzerovar[,-removecor]  
dim(training.decor)

## [1] 19622 46

### Split data to training and testing for cross validation.

inTrain <- createDataPartition(y=training.decor$classe, p=0.7, list=FALSE)  
training <- training.decor[inTrain,]; testing <- training.decor[-inTrain,]  
dim(training)

## [1] 13737 46

dim(testing)

## [1] 5885 46

### Analysis

#### Random Forests

Lets fit a random forest and see how well it performs

require(randomForest)  
set.seed(12345)  
rf.training=randomForest(classe~.,data=training,ntree=100, importance=TRUE)  
rf.training

##   
## Call:  
## randomForest(formula = classe ~ ., data = training, ntree = 100, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 100  
## No. of variables tried at each split: 6  
##   
## OOB estimate of error rate: 0.67%  
## Confusion matrix:  
## A B C D E class.error  
## A 3901 3 0 2 0 0.001280082  
## B 16 2633 7 0 2 0.009405568  
## C 0 19 2370 6 1 0.010851419  
## D 0 0 26 2222 4 0.013321492  
## E 1 1 2 2 2519 0.002376238

### Out-of Sample Accuracy

tree.pred=predict(rf.training,testing,type="class")  
predMatrix = with(testing,table(tree.pred,classe))  
sum(diag(predMatrix))/sum(as.vector(predMatrix))

## [1] 0.9945624

### Final prediction

Now we can predict the testing data provided.

prediction <- predict(rf.training, testingOrg)  
prediction

## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20   
## B A B A A E D B A A B C B A E E A B B B   
## Levels: A B C D E