Practical Machine Learning Course Project

Anna Harrington

5/10/2020

Introduction

For this project, my goal was to use data from accelerometers on the belt, forearm, arm and dumbbell for six study participants to quantify how well they performed barbell lifts. This report will go over how I built my machine learning model, how I used cross validation, my expected out of sample error, and the reasoning behind each. After performing this analysis, I will apply my prediction model to predict 20 different test cases.

Analysis

Cleaning the Data and Cross Validation

First, I loaded in the training and testing data provided in the instructions using the read.csv function and set the seed for the project as 1993 using the set.seed function. Then I removed the first seven columns of both data frames as they were not relevant to the exercise. Next, I cleaned up the data sets by removing all columns where the majority (more than 90%) of the data was NA values using the sapply, mean, and is.na functions. Next, I applied cross validation. The reasoning behind this is that cross validation allows me to evaluate the model with the limited data set that is provided in this assignment. I did this using the createDataPartition function. I split the trainingdata data frame using a p value of 0.7 to create two data frames- trainingdata and testingdatasplut. The code for this is shown below.

```
library(caret)

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

## Loading required package: ggplot2

library(corrplot)

## Warning: package 'corrplot' was built under R version 3.6.3

## corrplot 0.84 loaded

library(e1071)

## Warning: package 'e1071' was built under R version 3.6.3

library(randomForest)
```

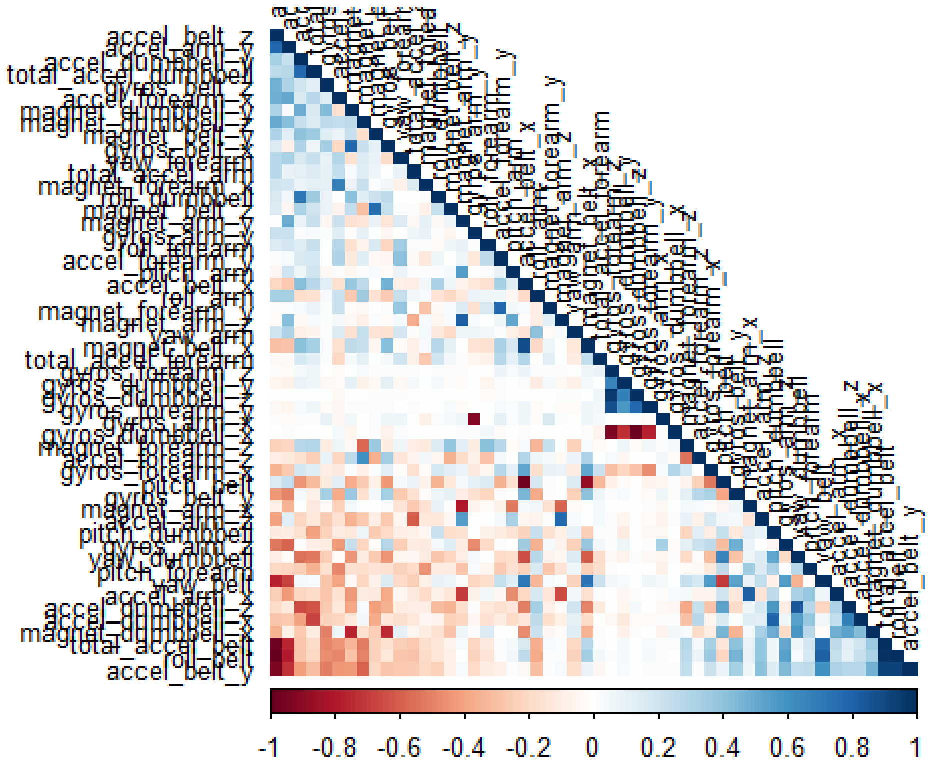
```
## Warning: package 'randomForest' was built under R version 3.6.3

## 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
```

```
set.seed(1993)
trainingdatain <-
read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"))
testingdatain  <-
read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"))

trainingdata <- trainingdatain
testingdata <- testingdatain

trainingdata <- trainingdata[, -c(1:7)]
testingdata <- testingdata[, -c(1:7)]
naremove    <- sapply(trainingdata, function(x) mean(is.na(x))) > 0.9
trainingdata <- trainingdata[, naremove==FALSE]
testingdata  <- testingdata[, naremove==FALSE]

intrain <- createDataPartition(trainingdata$classe, p=0.7, list=FALSE)
trainingdata <- trainingdata[intrain, ]
testingdatasplit <- trainingdata[-intrain, ]

near0variance <- nearZeroVar(trainingdata)
trainingdata <- trainingdata[, -near0variance]
testingdatasplit  <- testingdatasplit[, -near0variance]
```

Random Forest Model

Next I built a random forest model for the data. I chose to use the random forest modeling technique because according to my lecture notes from week three of this course, random forest is the most accurate. The cons of using random forest are the speed, interpretability, and overfitting; for this assignment, the accuracy outweighed all the listed cons.
First I used the cor and corrplot functions to create a plot showing the correlations between variables in the trainingdata data frame. Then I used the trainControl and train functions to build the random forest model (randomforest1). Next I used the predict and confusionMatrix functions to generate randomforest3.

```
correlation <- cor(trainingdata[, -53])

corrplot(correlation, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8 , tl.col = rgb(0, 0, 0))
```



```
randomforest <- trainControl(method="cv", number=3, verboseIter=FALSE)
randomforest1 <- train(classe ~., data=trainingdata, trControl=randomforest)
print(randomforest1$finalModel)

##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.74%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3896    8    1    0    1 0.002560164
## B   18 2634    6    0    0 0.009029345
## C    0   16 2372    8    0 0.010016694
## D    0    0   27 2221    4 0.013765542
## E    1    1    4    7 2512 0.005148515
```

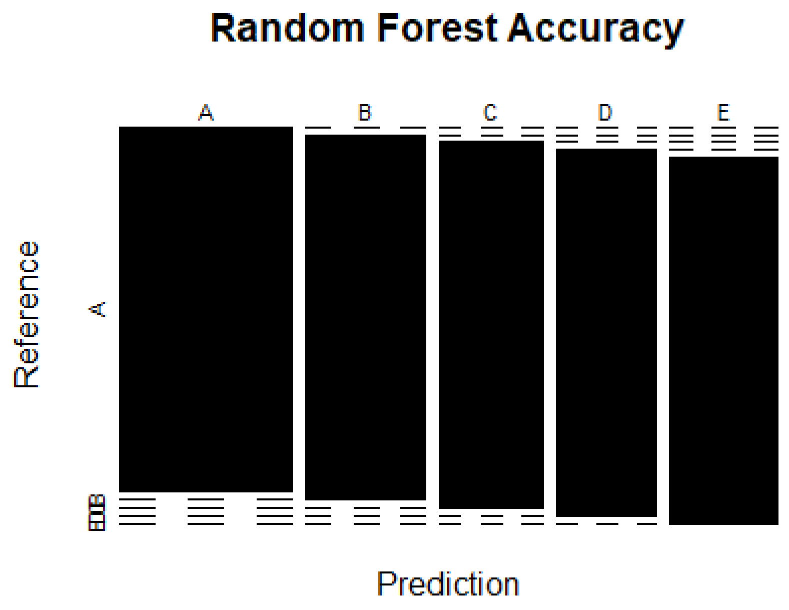
```
randomforest2 <- predict(randomforest1, newdata=testingdatasplit)
randomforest3 <- confusionMatrix(randomforest2, testingdatasplit$class)
print(randomforest3)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1171    0    0    0    0
##          B    0  817    0    0    0
##          C    0    0  707    0    0
##          D    0    0    0  679    0
##          E    0    0    0    0  729
## 
## Overall Statistics
##                                      
##                Accuracy : 1          
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2854     
##     P-Value [Acc > NIR] : < 2.2e-16  
##                                      
##                   Kappa : 1          
##                                      
##  Mcnemar's Test P-Value : NA         
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2854   0.1991   0.1723   0.1655   0.1777
## Detection Rate         0.2854   0.1991   0.1723   0.1655   0.1777
## Detection Prevalence   0.2854   0.1991   0.1723   0.1655   0.1777
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Expected Out of Sample Error

The plotted accuracy of the random forest model is shown below. According to the output the accuracy of the random forest model is 1. The fact that the value is so high could be due to overfitting (which is one of the problems with random forest models). Based on this the out of sample error for the random forest model is 0. The reasoning behind this is that the out of sample error is equal to one minus the accuracy of the model.

```
plot(randomforest3$table, col=randomforest3$byClass, main="Random Forest
Accuracy")
```



Results

For this assignment, I built a random forest model to predict how well six study participants performed barbell lifts using data from accelerometers placed on the belt, forearm, arm, and dumbbell. Cross validation was applied to allow for model evaluation. The expected out of sample error is 0 because the accuracy of the random forest model is 1 (although this may be due to overfitting). Finally, I applied the random forest model to the testing data set provided to generate results for the quiz portion of the assignment. The results are shown below.

```r
randomforest4 <- predict(randomforest1, newdata=testingdata)
print(randomforest4)

##  [1] 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
```