Practical Machine Learning Prediction Project

Robert Ramers

July 9, 2016

Introduction

The purpose of this project is to predict the manner in which users of personal activity devices such as Fitbit, Jawbone UP, and Nike FuelBand performed barbell fits using data from accelerometers on the belt, forearm, arm and dumbell.

The variable that contains the way in which they did the exercise is the "classe" variable. This report will describe how we built the model, how we used cross validation, and what we think the expected out of sample error is.

Reproducibility

To insure reproducibility, please make sure that you install and load the following packages and set the seed as shown below:

```
library (caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart)
library(rpart.plot)
library(randomForest)

## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

set.seed(12345)
```

Also, be sure to install the following in your environment: install.packages('e1071', dependencies=TRUE)

or when installing caret, use: install.packages('caret', dependencies = TRUE)

Getting and cleaning the data

The data required can be found here: Training data:

```
trainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
training <- read.csv(url(trainUrl),na.strings=c("NA","#DIV/0!",""))
```

Test/validation data:

```
testURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
testing <- read.csv(url(testURL), na.strings=c("NA","#DIV/0!",""))
```

Partitioning the data

Prediction of the classes will be done on the testing (validation) dataset, so we will split the training data into training and testing section. We an use the pml-testing.csv as a validation sample. We can then use cross validation within ght training partition to improve the fit of the model, and then can perform an out-of-sample test with the testing section.

```
goodTrain <- createDataPartition(training$classe, p=0.6, list=FALSE)
myTraining <- training[goodTrain, ]
myTesting <- training[-goodTrain, ]
dim(myTraining); dim(myTesting)

## [1] 11776   160

## [1] 7846  160
```

Eliminating zero features, first column of training set and variables with > 60% NA

To predict classes in the validation sample, we will eliminate all the fatures that are near 0 in the validation set.

```
nzv <- nearZeroVar(myTraining, saveMetrics=TRUE)

myTraining <- myTraining[,nzv$nzv==FALSE]

nzv<- nearZeroVar(myTesting,saveMetrics=TRUE)

myTesting <- myTesting[,nzv$nzv==FALSE]
```

Remove first column of myTraining data

```
myTraining <- myTraining[c(-1)]
```

Clean variables with >0% NA

```
trainingA <- myTraining
for(i in 1:length(myTraining)) {
    if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .7) {
        for(j in 1:length(trainingA)) {
            if( length( grep(names(myTraining[i]), names(trainingA)[j]) ) ==
1)  {
                trainingA <- trainingA[ , -j]
            }
        }
    }
}



# reset initial variable name
myTraining <- trainingA
rm(trainingA)
```

Transform the data sets

```
cleana <- colnames(myTraining)
cleanb <- colnames(myTraining[, -58])  # remove the classe column
myTesting <- myTesting[cleana]         # allow only variables in myTesting
that are also in myTraining
testing <- testing[cleanb]            # allow only variables in testing that
are also in myTraining


dim(myTesting)

## [1] 7846   58

dim(testing)

## [1] 20 57

# Coerce data into the same type

for (i in 1:length(testing) ) {
    for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) == 1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}


testing <- rbind(myTraining[2, -58] , testing)
testing <- testing[-1,]
```

Building the models

We will test three models and then determine which on has the best out-of-sample accuracy.

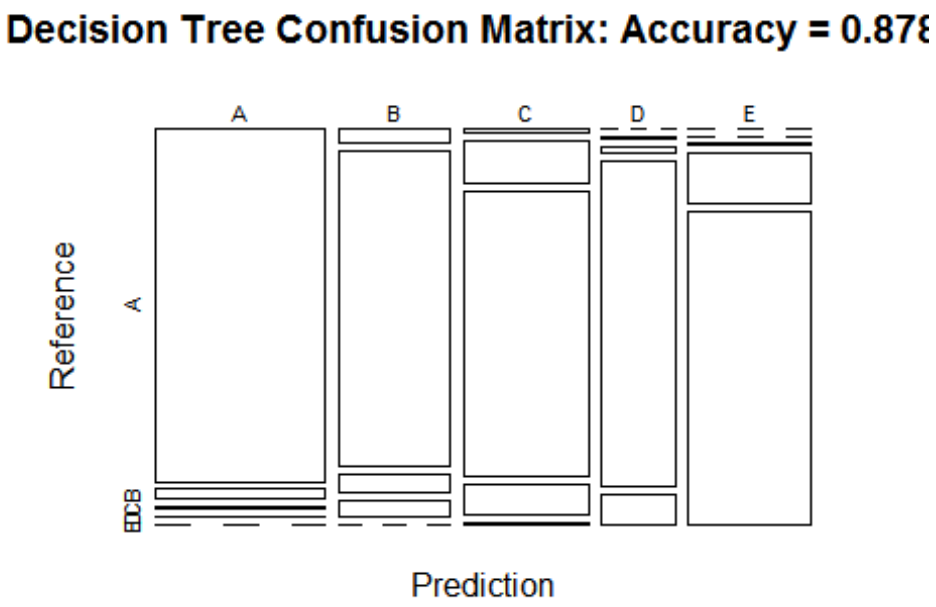
These models are: 1. Decision trees with CART (rpart) 2. Random forest decision trees (rf)

Decision Tree

```
modFitDT <- rpart(classe ~ ., data=myTraining, method="class")
predictionsDT <- predict(modFitDT, myTesting, type = "class")
cmtree <- confusionMatrix(predictionsDT, myTesting$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2150   60    7    1    0
##          B   61 1260   69   64    0
##          C   21  188 1269  143    4
##          D    0   10   14  857   78
##          E    0    0    9  221 1360
##
## Overall Statistics
##
##                Accuracy : 0.8789
##                  95% CI : (0.8715, 0.8861)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8468
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9633   0.8300   0.9276   0.6664   0.9431
## Specificity            0.9879   0.9693   0.9450   0.9845   0.9641
## Pos Pred Value         0.9693   0.8666   0.7809   0.8936   0.8553
## Neg Pred Value         0.9854   0.9596   0.9841   0.9377   0.9869
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2740   0.1606   0.1617   0.1092   0.1733
## Detection Prevalence   0.2827   0.1853   0.2071   0.1222   0.2027
## Balanced Accuracy      0.9756   0.8997   0.9363   0.8254   0.9536
```

```
plot(cmtree$table, col = cmtree$byClass, main = paste("Decision Tree
Confusion Matrix: Accuracy =", round(cmtree$overall['Accuracy'], 4)))
```



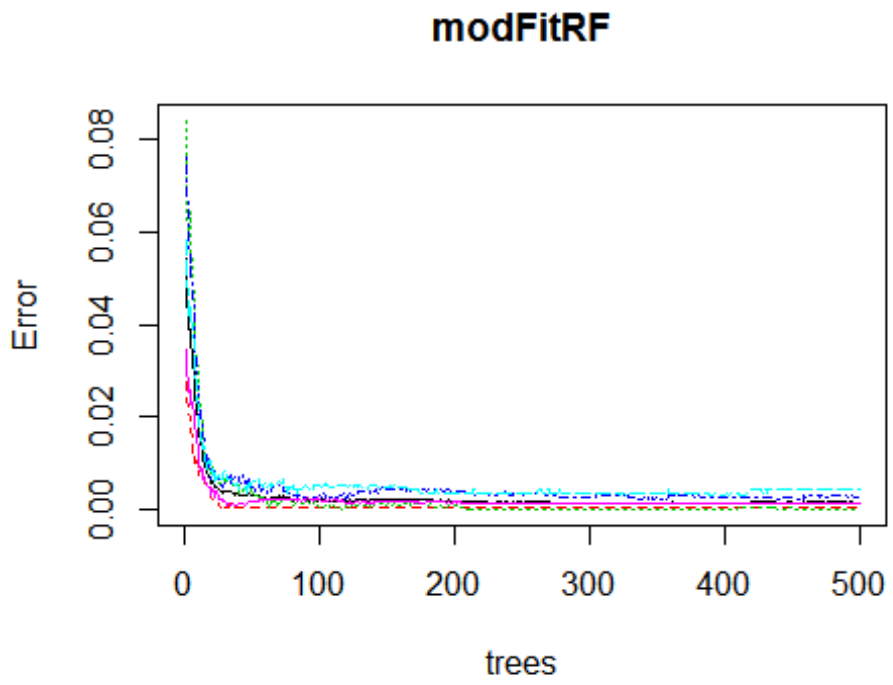
Random Forest

```
set.seed(12345)
modFitRF <- randomForest(classe ~ ., data=myTraining)
predictionRF <- predict(modFitRF, myTesting, type = "class")
cmrf <- confusionMatrix(predictionRF, myTesting$classe)
cmrf
```

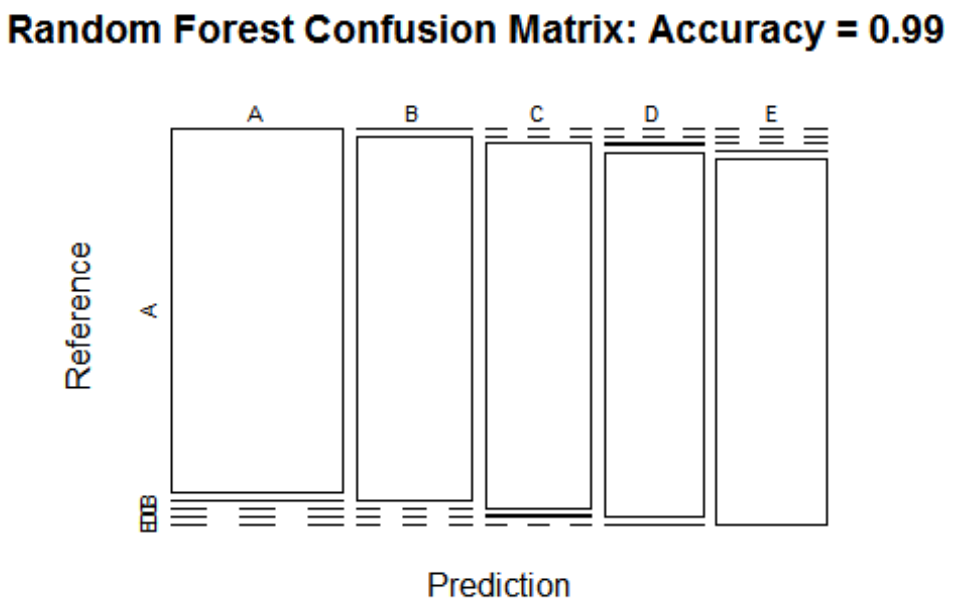
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    2    0    0    0
##          B    1 1516    0    0    0
##          C    0    0 1367    3    0
##          D    0    0    1 1282    1
##          E    0    0    0    1 1441
##
## Overall Statistics
##
##                Accuracy : 0.9989
##                  95% CI : (0.9978, 0.9995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9987   0.9993   0.9969   0.9993
## Specificity            0.9996   0.9998   0.9995   0.9997   0.9998
## Pos Pred Value         0.9991   0.9993   0.9978   0.9984   0.9993
## Neg Pred Value         0.9998   0.9997   0.9998   0.9994   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1742   0.1634   0.1837
## Detection Prevalence   0.2846   0.1933   0.1746   0.1637   0.1838
## Balanced Accuracy      0.9996   0.9993   0.9994   0.9983   0.9996
```

```
plot(modFitRF)
```




```
plot(cmrf$table, col = cmtree$byClass, main = paste("Random Forest Confusion
Matrix: Accuracy =", round(cmrf$overall['Accuracy'], 4)))
```



Predicting results with the test data

Random Forest provided a greater accuracy in the testing data than the Decision Tree.

```
predictionRF <- predict(modFitRF, testing, type = "class")
predictionRF
```

```
##  2 31  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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
```

Conclusion

It is clear that the random forest model produces a very accurate prediction.