Practical Machine Learning Assignment

Davayne Melbourne

June 12, 2016

Introduction

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 - 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 project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They 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).

Data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Purpose

The goal of the project is to use a predition algorithm and use data from accelerometers on the belt, forearm, arm, and dumbell to predict whetehr or not a particular activity was performed correctly. This is the "classe" variable in the training set. After developing a prediction model, the prediction will be applied to 20 test cases for evaluation.

Loading and Cleaning Data

Loading

```
trainingbase <- read.csv("pml-training.csv", na.strings=c("", "NA", "NULL","#DIV/0!"))
testingbase <- read.csv("pml-testing.csv", na.strings=c("", "NA", "NULL","#DIV/0!"))
dim(trainingbase)
```

```
## [1] 19622   160
```

```
dim(testingbase)
```

```
## [1]  20 160
```

remove unnecessary or identification variables

```
myvars <- names(trainingbase) %in% c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
trainingred <- trainingbase[!myvars]
dim(trainingred)
```

```
## [1] 19622   153
```

remove variables with mainly NA

```
trainingred1 <- trainingred[ , colSums(is.na(trainingred)) == 0]
dim(trainingred1)
```

```
## [1] 19622    53
```

remove variables with near zero variance

```
nearzerovariance <- nearZeroVar(trainingred1, saveMetrics=TRUE)
trainingred2 <- trainingred1[, nearzerovariance$nzv==FALSE]
dim(trainingred2)
```

```
## [1] 19622    53
```

next we remove variables that are highly correlated with the independent variable

```
corrMatrix <- cor(na.omit(trainingred2[sapply(trainingred2, is.numeric)]))
highcorr <- findCorrelation(corrMatrix, cutoff = .90, verbose = TRUE)
```

```
## Compare row 10  and column  1 with corr  0.992
##   Means:  0.27 vs 0.168 so flagging column 10
## Compare row 1  and column  9 with corr  0.925
##   Means:  0.25 vs 0.164 so flagging column 1
## Compare row 9  and column  4 with corr  0.928
##   Means:  0.233 vs 0.161 so flagging column 9
## Compare row 8  and column  2 with corr  0.966
##   Means:  0.245 vs 0.157 so flagging column 8
## Compare row 19  and column  18 with corr  0.918
##   Means:  0.091 vs 0.158 so flagging column 18
## Compare row 46  and column  31 with corr  0.914
##   Means:  0.101 vs 0.161 so flagging column 31
## Compare row 46  and column  33 with corr  0.933
##   Means:  0.083 vs 0.164 so flagging column 33
## All correlations <= 0.9
```

```
trainingred3 <- trainingred2[,-highcorr]
dim(trainingred3)
```

```
## [1] 19622    46
```

Split the base training data into training and testing sets for cross validation.

```
set.seed(1000)
inTrain <- createDataPartition(y=trainingred3$classe, p=0.7, list=FALSE)
training <- trainingred3[inTrain,]
testing <- trainingred3[-inTrain,]
dim(training); dim(testing)
```

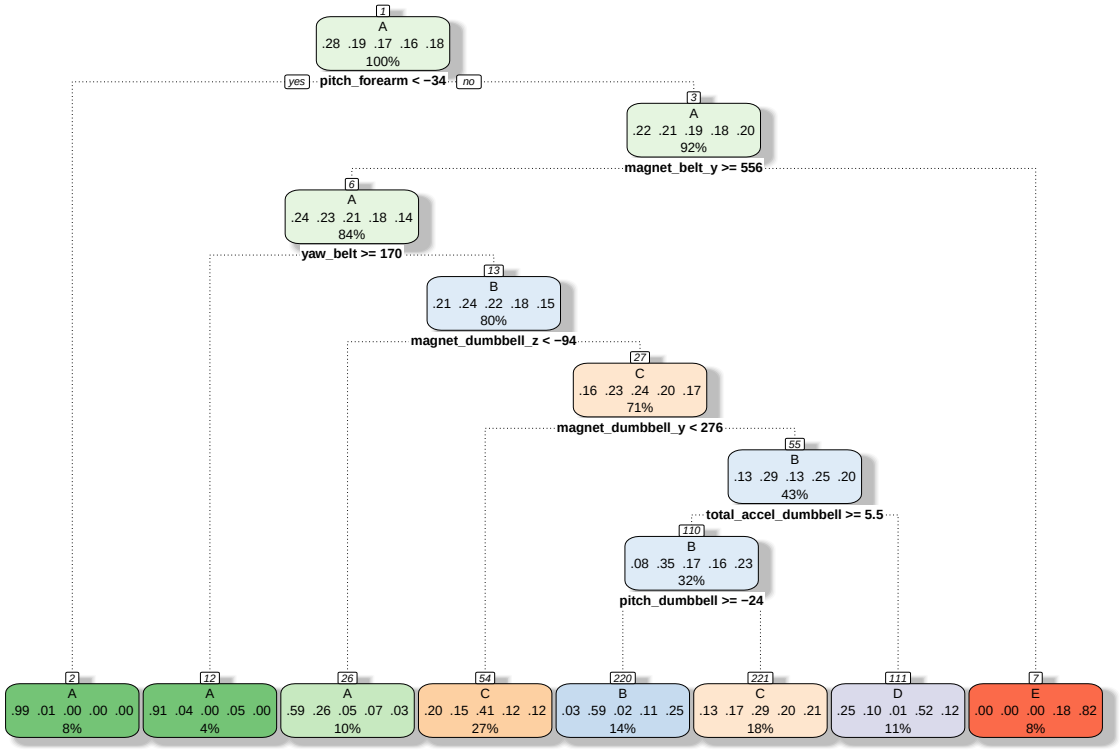
```
## [1] 13737    46
```

```
## [1] 5885   46
```

Prediction

We fit a decision tree to the data then evaluate accuarcy.

```
modFit1 <- train(classe ~ .,method="rpart",data=training)
fancyRpartPlot(modFit1$finalModel)
```



Cross Validation We check the performance of the tree on the testing set for cross validation.

```
predictions1 <- predict(modFit1, testing, type = "raw")
cmtr <- confusionMatrix(predictions1, testing$classe)
cmtr
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   A   B   C   D   E
##          A 991 196  27  32  17
##          B  42 490  23  90 205
##          C 452 396 971 460 430
##          D 187  56   5 300  83
##          E   2   1   0  82 347
##
## Overall Statistics
##
##                Accuracy : 0.5266
##                  95% CI : (0.5137, 0.5394)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.408
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5920  0.43020   0.9464  0.31120  0.32070
## Specificity            0.9354  0.92415   0.6423  0.93274  0.98230
## Pos Pred Value         0.7846  0.57647   0.3584  0.47544  0.80324
## Neg Pred Value         0.8522  0.87110   0.9827  0.87362  0.86521
## Prevalence             0.2845  0.19354   0.1743  0.16381  0.18386
## Detection Rate         0.1684  0.08326   0.1650  0.05098  0.05896
## Detection Prevalence   0.2146  0.14444   0.4603  0.10722  0.07341
## Balanced Accuracy      0.7637  0.67717   0.7944  0.62197  0.65150
```

From Cross validation we see that the out of sample accuracy of tree is low so we will try other methods.

Random Forest

Lets fit a random forest (using average of multiple trees) and evaluate well it performs.

```
set.seed(1000)
modFit2 = randomForest(classe~.,data=training)
modFit2
```

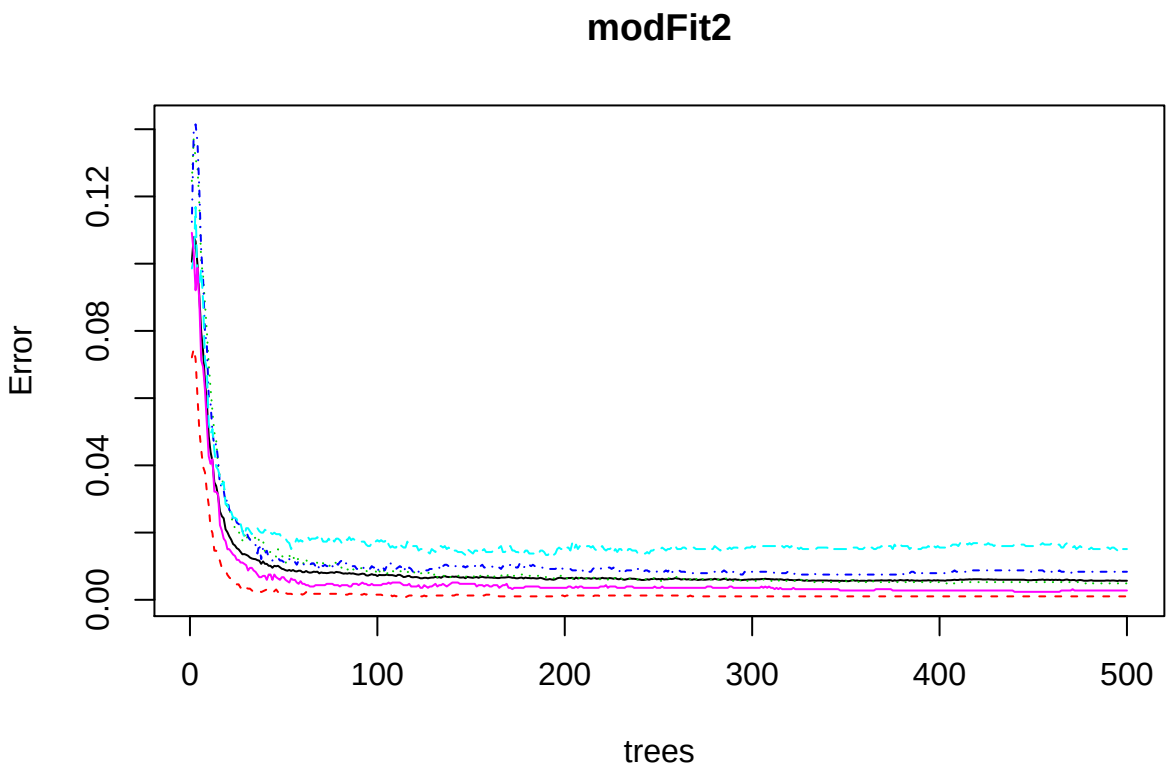
```
##
## Call:
##  randomForest(formula = classe ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 0.57%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    3    0    0    1 0.001024066
## B    9 2645    4    0    0 0.004890895
```

```
## C    0   19 2376    1    0 0.008347245
## D    0    0   29 2218    5 0.015097691
## E    0    0    1    6 2518 0.002772277
```

```
predictions2 <- predict(modFit2, testing, type = "class")
cmrf <- confusionMatrix(predictions2, testing$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    6    0    0    0
##          B    3 1132    7    0    0
##          C    0    1 1017   10    0
##          D    0    0    2  953    3
##          E    0    0    0    1 1079
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9921, 0.9961)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9939   0.9912   0.9886   0.9972
## Specificity            0.9986   0.9979   0.9977   0.9990   0.9998
## Pos Pred Value         0.9964   0.9912   0.9893   0.9948   0.9991
## Neg Pred Value         0.9993   0.9985   0.9981   0.9978   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1924   0.1728   0.1619   0.1833
## Detection Prevalence   0.2850   0.1941   0.1747   0.1628   0.1835
## Balanced Accuracy      0.9984   0.9959   0.9945   0.9938   0.9985
```

```
plot(modFit2)
```



Overall, the Random forest method has great out of sample accuracy of 99.4% and so we will use as our prediction algorithm. The plot indicates we needed only 50 tree repitititons to get good accuracy.

Conclusion

```
finalanswers <- predict(modFit2, testingbase, type = "class")
finalanswers
```

```
##  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
```