Course Project: Prediction

Chris Peck

December 21, 2016

Description

The goal of this project is to create an algorithm to identify how well participants performed dumbbbell bicep curls.

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset). The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Load the data

Load the training and test data sets

```
filetrain<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

filetest<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

traindata <- read.csv(file=filetrain, header=TRUE, sep=",")
testdata <- read.csv(file=filetest, header=TRUE, sep=",")
```

Clean the data

After viewing the data, it was evident that many fields contained blanks and NAs for the majority of the measurements (19,216 out of 19.622 observations). These fields were eliminated from the data we use since they can’t be used in the prediction algorithm and to help speed up the calculations on the data set. We do not show the head(traindataclean) to conserve space in the report.

```r
##convert blanks to NA
traindatana <- read.csv(file=filetrain, header=TRUE, sep=",",na.strings=c(""," ","NA"))
na_count <-sapply(traindatana, function(y) sum(length(which(is.na(y)))))
na_count <- data.frame(na_count)
##na_count

##we see 19,216 NAs for many of the variables out of the 19,622 observations so we remove these
 columns to reduce the number of variables for our model fitting.  Statistics not shown to conse
rve space in the report.
traindataclean<-traindatana[ , ! apply( traindatana , 2 , function(x) any(is.na(x)) ) ]

##remove the first 7 columns which aren't part of the measurements taken by the monitor
traindataclean<-traindataclean[,c(8:60)]

##we can see there are no more variables that are blank or NA
##head(traindataclean)
```

Split the training data into a training set and a test set

We split the training data into a training subset and a test subset so that we can do a cross-validation of the model prior to applying it to the 20 samples in the test data

```r
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```r
set.seed(1234)
trainIndex <- createDataPartition(traindataclean$classe, p=0.7, list=FALSE)
data_train_train_subset <- traindataclean[ trainIndex,]
data_train_test_subset <- traindataclean[-trainIndex,]
```

Create various models for prediction

rpart

Try rpart model for prediction

```r
install.packages("rattle")
```

```
## package 'rattle' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\cpeck\AppData\Local\Temp\RtmpmkiJpl\downloaded_packages
```

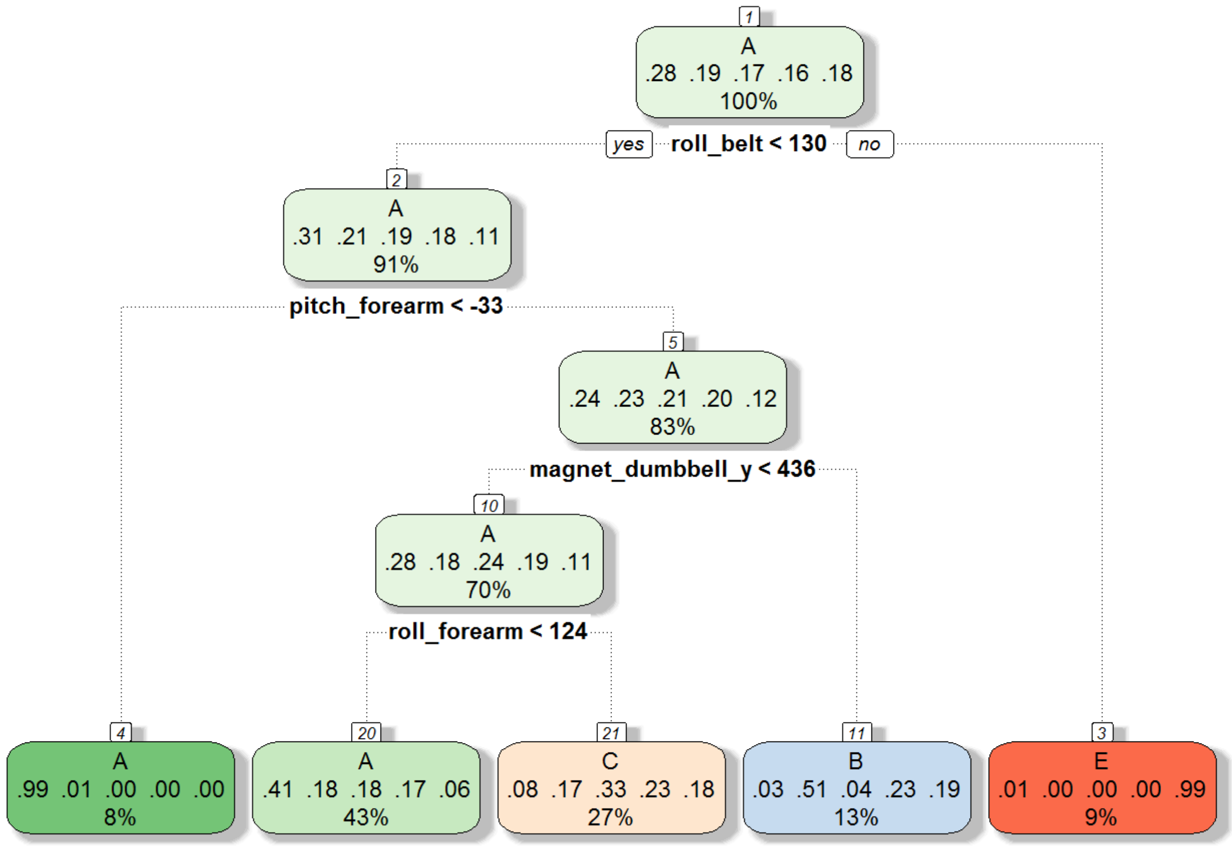
```r
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
modelfittree<-train(classe~.,method="rpart", data=data_train_train_subset, control = rpart.contr
ol(maxdepth = 5))
```

```
## Loading required package: rpart
```

```
fancyRpartPlot(modelfittree$finalModel)
```



Rattle 2017-Jan-01 20:42:44 cpeck

Try predicting with the rpart model. We can see it doesn’t work very well based on the cross-validation table, which contains many misclassifications in the predtree variable compared to the actual classe.

```
predtree<-predict(modelfittree,data_train_test_subset)
table(predtree,data_train_test_subset$classe)
```

```
##
## predtree    A    B    C    D    E
##        A 1530  486  493  452  168
##        B   35  379   31  164  145
##        C  105  274  502  348  302
##        D    0    0    0    0    0
##        E    4    0    0    0  467
```

Random Forest

Try random forest model. We can see that this works well based on the confusion matrix, which has very low class errors with fashion D having the greatest error rate of 1.3%. The variable importance plot shows that the yaw belt, roll belt, magnet dumbbell z, magnet dumbbell y and pitch belt are the most important variables in predicting the classe.

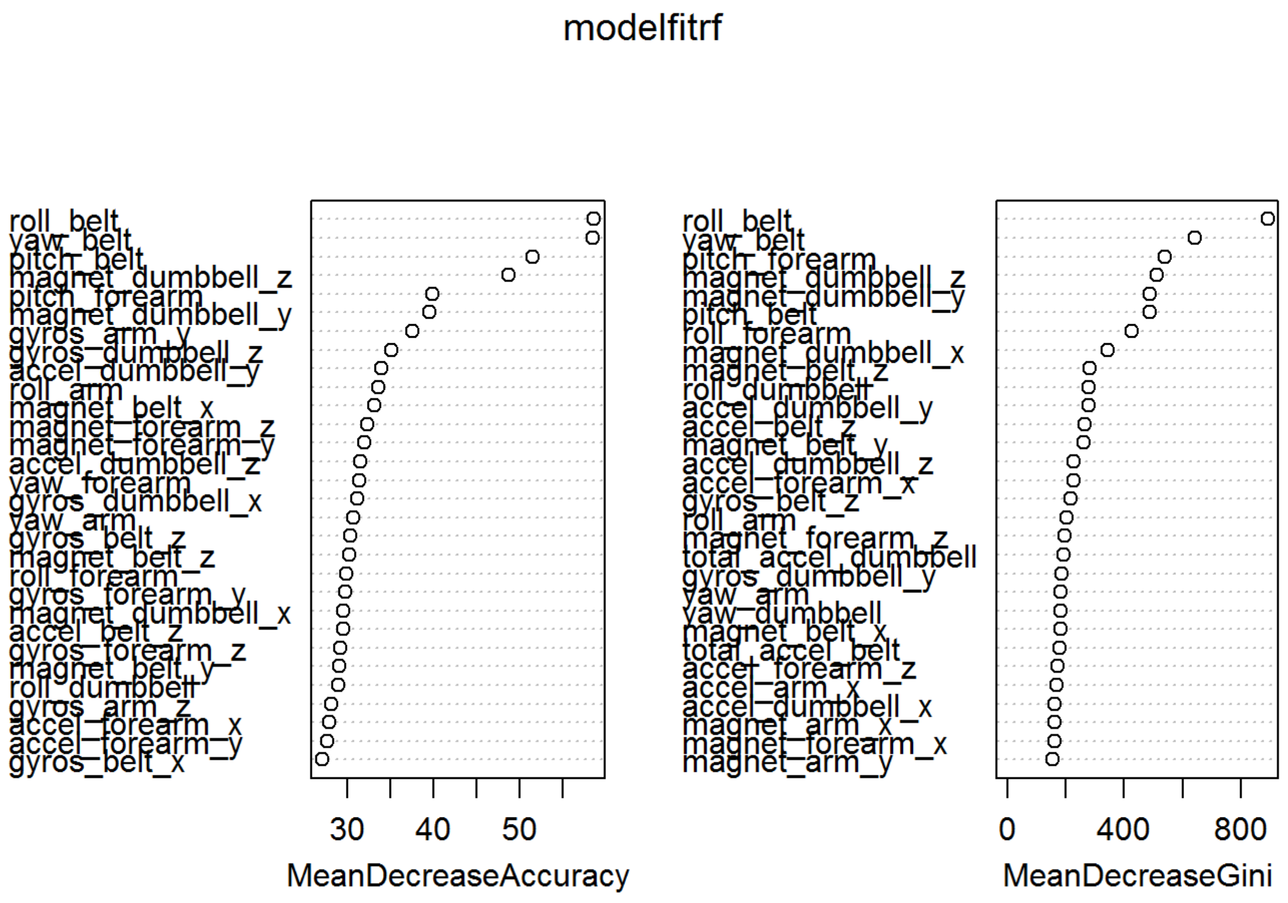
```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
modelfitrf <- randomForest(classe ~ ., data=data_train_train_subset, importance=TRUE)
modelfitrf
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = data_train_train_subset,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.56%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3903    3    0    0    0 0.0007680492
## B   13 2642    3    0    0 0.0060195636
## C    0   16 2376    4    0 0.0083472454
## D    0    0   28 2222    2 0.0133214920
## E    0    0    1    7 2517 0.0031683168
```

```
varImpPlot(modelfitrf)
```



Try predicting with the random forest on the training test data. We can see it works well as there are very few predictions that don’t match the actual classe in the confusion matrix.

```
predrf<-predict(modelfitrf,data_train_test_subset)
predtab<-table(predrf,data_train_test_subset$classe)
confusionMatrix(predtab)
```

```
## Confusion Matrix and Statistics
##
##
## predrf    A    B    C    D    E
##      A 1674    8    0    0    0
##      B    0 1130    6    0    0
##      C    0    1 1020    4    0
##      D    0    0    0  959    0
##      E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.9948, 0.9979)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9957
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9921   0.9942   0.9948   1.0000
## Specificity            0.9981   0.9987   0.9990   1.0000   0.9998
## Pos Pred Value         0.9952   0.9947   0.9951   1.0000   0.9991
## Neg Pred Value         1.0000   0.9981   0.9988   0.9990   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1920   0.1733   0.1630   0.1839
## Detection Prevalence   0.2858   0.1930   0.1742   0.1630   0.1840
## Balanced Accuracy      0.9991   0.9954   0.9966   0.9974   0.9999
```

Expected out of sample error

From the confusion matrix above, we can see that the out of sample error of 0.3% is very low based on the accuracy of 99.7%. Out of sample error is calculated as 1 - accuracy.

Conclusion

I chose the random forest model because it produces a high level of accuracy, works very well in predicting using the test portion of the training data set (as seen from confusion matrix) and runs quickly enough in R.

Course Project Prediction Quiz Portion: Predict on the test data using Random Forest Model

We apply the random forest model to predict the classe of the 20 test observations for the quiz poriton of the project.

```r
predrftest<-predict(modelfitrf,testdata)
predrftest
```

```
##  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
```