# Barbell Lift Classification

Phil Renner

12/17/2021

## 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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

## Setup and Reading Data

```
knitr::opts_chunk$set(echo = TRUE)


library(caret)
library(knitr)
library(ggplot2)
library(lattice)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
library(corrplot)
```

```
library(gbm)
library(kernlab)
```

```
#read into data frame
validation <- read.csv("pml-testing.csv")
training <-  read.csv("pml-training.csv")
#dim(validation)
#dim(training)

#clean up data by removing variables with missing data

training<- training[, colSums(is.na(training)) == 0]
validation <- validation[, colSums(is.na(validation)) == 0]
#dim(training)
#dim(validation)


#remove first seven variables, as they are descriptive and don't influence classe
training <- training[,-c(1:7)]

#remove variables with near zero variance
nvz <- nearZeroVar(training)
training <- training[,-nvz]
#dim(training)

#divide into training and testing set
inTrain <- createDataPartition(y=training$classe, p=0.7, list=F)
train <- training[inTrain,]
testing <- training[-inTrain,]
```

We will try several different prediction approaches. We will use Decision Trees, Random Forest, and SVM. I will use 3-fold cross-validation

Decision Tree Model:

```
set.seed(54321)
control <- trainControl(method="cv", number=3, verboseIter=F)

mod_trees <- train(classe~., data=train, method="rpart", trControl = control, tuneLength = 5)
#fancyRpartPlot(mod_trees$finalModel)

# Produce confusion matrix for decision tree model
pred_trees <- predict(mod_trees, testing)
cmtrees <- confusionMatrix(pred_trees, factor(testing$classe))
cmtrees
```

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

```
## Prediction    A    B    C    D    E
##          A 1517  472  509  430  160
##          B   20  290   15  165   66
##          C  100  118  404  138  149
##          D   16   44   10  175   33
##          E   21  215   88   56  674
##
## Overall Statistics
##
##                Accuracy : 0.52
##                  95% CI : (0.5071, 0.5328)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3724
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9062  0.25461  0.39376  0.18154   0.6229
## Specificity            0.6269  0.94395  0.89607  0.97907   0.9209
## Pos Pred Value         0.4913  0.52158  0.44444  0.62950   0.6395
## Neg Pred Value         0.9439  0.84068  0.87500  0.85928   0.9155
## Prevalence             0.2845  0.19354  0.17434  0.16381   0.1839
## Detection Rate         0.2578  0.04928  0.06865  0.02974   0.1145
## Detection Prevalence   0.5247  0.09448  0.15446  0.04724   0.1791
## Balanced Accuracy      0.7666  0.59928  0.64492  0.58030   0.7719
```

The results of the decision tree model show accuracy = 0.6477 and Out of Sample error is 0.3523.

Random Forest Model

```
set.seed(54321)
control <- trainControl(method="cv", number=3, verboseIter=F)

mod_forest <- train(classe~., data=train, method="rf", trControl = control, tuneLength = 5)


# Produce confusion matrix for random forest model
pred_forest <- predict(mod_forest, testing)
cmforest <- confusionMatrix(pred_forest, factor(testing$classe))
cmforest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    5    0    0    0
##          B    2 1133    4    0    0
##          C    0    1 1020   13    2
##          D    0    0    2  951    4
##          E    1    0    0    0 1076
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9942
##                  95% CI : (0.9919, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9927
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9947   0.9942   0.9865   0.9945
## Specificity            0.9988   0.9987   0.9967   0.9988   0.9998
## Pos Pred Value         0.9970   0.9947   0.9846   0.9937   0.9991
## Neg Pred Value         0.9993   0.9987   0.9988   0.9974   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1925   0.1733   0.1616   0.1828
## Detection Prevalence   0.2848   0.1935   0.1760   0.1626   0.1830
## Balanced Accuracy      0.9985   0.9967   0.9954   0.9926   0.9971
```

Accuracy is much better with the random forest model. Accuracy is 0.9932 and an out of sample error of 0,0068.

Support Vector Machine Model

```
set.seed(54321)
control <- trainControl(method="cv", number=3, verboseIter=F)

mod_svm <- train(classe~., data=train, method="svmLinear", trControl = control, tuneLength = 5, verbose


# Produce confusion matrix for SVM model
pred_svm <- predict(mod_svm, testing)
cmsvm <- confusionMatrix(pred_svm, factor(testing$classe))
cmsvm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1528  158   89   69   62
##          B   23  823   92   45  143
##          C   60   64  798   98   82
##          D   56   21   33  708   55
##          E    7   73   14   44  740
##
## Overall Statistics
##
##                Accuracy : 0.7811
##                  95% CI : (0.7704, 0.7916)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7217
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9128   0.7226   0.7778   0.7344   0.6839
## Specificity            0.9102   0.9362   0.9374   0.9665   0.9713
## Pos Pred Value         0.8017   0.7309   0.7241   0.8110   0.8428
## Neg Pred Value         0.9633   0.9336   0.9523   0.9489   0.9317
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2596   0.1398   0.1356   0.1203   0.1257
## Detection Prevalence   0.3239   0.1913   0.1873   0.1483   0.1492
## Balanced Accuracy      0.9115   0.8294   0.8576   0.8505   0.8276
```

The SVM model has an accuracy of 0.7791 and an out of sample error of 0.2209

The SVM model performs better than the Decision Tree model, but not as good as the Random Forest model. I'll use the Random Forest model to predict the 20 cases in the validation set.

```
#set.seed(54321)
#control <- trainControl(method="cv", number=3, verboseIter=F)

#mod_forest <- train(classe~., data=train, method="rf", trControl = control, tuneLength = 5)


# Produce confusion matrix for random forest model
pred_val <- predict(mod_forest, validation)
pred_val
```

```
##  [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
```