Practical Machine Learning - Course Project

Andy Pulaski

August 7, 2016

Project 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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset). input data

Project Goal

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Input Data

```
training <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv')

testing <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv')
```

Prep environment

```
set.seed(94954)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.2.5
```

```
library(rpart)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.2.5
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

Data Preparation

Partition the training set into training and validation sets

```
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
ptrain <- training[inTrain,]
ptest <- training[-inTrain,]
```

Now remove variables with nearly zero variance and predominately missing values in the training set

```
# show original size of training set
dim(ptrain)
```

```
## [1] 11776   160
```

```
nzv <- nearZeroVar(ptrain)
ptrain <- ptrain[, -nzv]
ptest <- ptest[, -nzv]

# remove vars with > 80% NA values
highNA <- sapply(ptrain, function(x) mean(is.na(x))) > .80
ptrain <- ptrain[,highNA==F]
ptest <- ptest[,highNA==F]

# remove the first five columns which are not useful predictive variables (nam
e & timestamps)
ptrain <- ptrain[,-(1:5)]
ptest <- ptest[,-(1:5)]

#show the new size of the training set after removing variables
dim(ptrain)
```

```
## [1] 11776    54
```

Modeling

Use the following modeling functions:

1. Model with a decision tree (RPART)
2. Model with random forest (RF)
3. Model with gradient boosting tree (GBM)

```
modFitTree <- train(classe~., data=ptrain, method='rpart')
modFitRF <- train(classe~., data=ptrain, method='rf')
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
## 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
```

```
modFitGBM <- train(classe~., data=ptrain, method='gbm')
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.2.5
```

```
## Loading required package: survival
```

```
## Warning: package 'survival' was built under R version 3.2.5
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

```
## Loading required package: plyr
```

```
## Warning: package 'plyr' was built under R version 3.2.5
```

Model Evaluation

Now predict each model on the validation set to allow us to re-train the models if necessary before the final test on the test set.

```
predTree <- predict(modFitTree, newdata = ptest)
predRF <- predict(modFitRF, newdata = ptest)
predGBM <- predict(modFitGBM, newdata = ptest)
```

results for decision tree

```
confusionMatrix(ptest$classe, predTree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2042   36  150    0    4
##          B  630  525  363    0    0
##          C  619   33  716    0    0
##          D  526  220  476    0   64
##          E  145  113  293    0  891
##
## Overall Statistics
##
##                Accuracy : 0.532
##                  95% CI : (0.5209, 0.5431)
##     No Information Rate : 0.505
##     P-Value [Acc > NIR] : 8.869e-07
##
##                   Kappa : 0.3895
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5154  0.56634  0.35836       NA   0.9291
## Specificity            0.9511  0.85648  0.88851   0.8361   0.9200
## Pos Pred Value         0.9149  0.34585  0.52339       NA   0.6179
## Neg Pred Value         0.6580  0.93647  0.80210       NA   0.9894
## Prevalence             0.5050  0.11815  0.25465   0.0000   0.1222
## Detection Rate         0.2603  0.06691  0.09126   0.0000   0.1136
## Detection Prevalence   0.2845  0.19347  0.17436   0.1639   0.1838
## Balanced Accuracy      0.7332  0.71141  0.62343       NA   0.9245
```

results for random forest

```
confusionMatrix(ptest$classe, predRF)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    0    0    0    1
##          B   11 1506    1    0    0
##          C    0    1 1367    0    0
##          D    0    0    2 1284    0
##          E    0    3    0    0 1439
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9976          
##                  95% CI : (0.9962, 0.9985)
##     No Information Rate : 0.2858          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9969          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9951   0.9974   0.9978   1.0000   0.9993
## Specificity            0.9998   0.9981   0.9998   0.9997   0.9995
## Pos Pred Value         0.9996   0.9921   0.9993   0.9984   0.9979
## Neg Pred Value         0.9980   0.9994   0.9995   1.0000   0.9998
## Prevalence             0.2858   0.1925   0.1746   0.1637   0.1835
## Detection Rate         0.2843   0.1919   0.1742   0.1637   0.1834
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9975   0.9977   0.9988   0.9998   0.9994
```

results for GBM

```
confusionMatrix(ptest$classe, predGBM)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2219   12    0    1    0
##          B   28 1474   14    2    0
##          C    0    3 1363    2    0
##          D    2    7   25 1251    1
##          E    2    1    0    8 1431
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9862          
##                  95% CI : (0.9834, 0.9887)
##     No Information Rate : 0.2869          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9826          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9858   0.9846   0.9722   0.9897   0.9993
## Specificity            0.9977   0.9931   0.9992   0.9947   0.9983
## Pos Pred Value         0.9942   0.9710   0.9963   0.9728   0.9924
## Neg Pred Value         0.9943   0.9964   0.9940   0.9980   0.9998
## Prevalence             0.2869   0.1908   0.1787   0.1611   0.1825
## Detection Rate         0.2828   0.1879   0.1737   0.1594   0.1824
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9917   0.9889   0.9857   0.9922   0.9988
```

Re-train Model

because the Random Forest produced a result with 99.7% accuracy on test set, I don't think we need to try to retrain the model with additional algorithms. At this point we will move straight to predicting the results on the 20 observation data set from our original input using the Random Forest model we created above.

```
predTestRF <- predict(modFitRF, newdata = testing)

predTestRF
```

```
##  [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
```