Practical Machine Learning - Course Project Week 4

Introduction

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).

Data

Load the data (and packages) and replace missing variables and #DIV/0 with NA

```
require("RCurl")
require("caret")

url1 <- getURL("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.c
sv")
training <- read.csv(text=url1,na.strings=c("NA","#DIV/0!", "") )

url2 <- getURL("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.cs
v")
testing <- read.csv(text=url2,na.strings=c("NA","#DIV/0!", "") )
```

Pre-processing

We want to remove some of this data, as it has no predictive role:

```
varToRemove <- names(training) %in% c("X", "user_name", "raw_timestamp_part_1", "r
aw_timestamp_part_2","cvtd_timestamp", "new_window", "new_window", "num_window")

training <- training[!varToRemove]
```

Let's also remove columns from the training set where there is too much missing data (say 80%+)

```
training <- training[colSums(!is.na(training))> nrow(training)*0.2]
```

That leaves us with 53 variables, including the 'classe' variable that we will be predicting Now, let's also exclude highly correlated varibles (see Jason Brownlee's blog on that here: http://machinelearningmastery.com/feature-selection-with-the-caret-r-package/ (http://machinelearningmastery.com/feature-selection-with-the-caret-r-package/))

```
library(caret) # Model package
correlationMatrix <- cor(training[, 1:52])
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.75)
training <- training[-c(highlyCorrelated)]
```

To identify the best model, we will build a couple of different once that we have come across during this coursera course: Tree based classification; Bagging; Random Forest, Support Vector Machine. First, however, we will create a hold-out sample, consisting of 30% of the training set.

```
set.seed(100)
trainIndex <- createDataPartition(training$classe, p=.7, list=FALSE)
training <- training[trainIndex,]
holdout <- training[-trainIndex,]
```

Model build

We are going to use the trainControl option to speed up computation of the models

```
tc <- trainControl( method = "cv", repeats = 5)
```

```
library(e1071) # Needed for rpart/tree model
library(ipred) # Needed for bagging
library(gbm) # Needed for boosting
library(randomForest) # Needed for Random Forest

set.seed(100)
treeModel <- train(classe ~., method="rpart", trControl=tc, data=training)
bagModel <-train(classe ~.,method="treebag", trControl=tc, data=training)
boostModel <- train(classe ~., method="gbm", trControl=tc, verbose=FALSE, data=tra
ining)
rfModel <- train(classe ~., method="rf", trControl=tc, data=training)
```

Now, let's have a look at the confusionMatrix's outputs to se what model is predicting best on the training sample

```
confusionMatrix(training$classe, predict(treeModel,training))$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.5493194      0.4326808      0.5409528      0.5576651      0.2705103
## AccuracyPValue  McnemarPValue
##      0.0000000      0.0000000
```

```
confusionMatrix(training$classe, predict(bagModel,training))$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9998544      0.9998159      0.9994742      0.9999824      0.2843416
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
confusionMatrix(training$classe, predict(boostModel,training))$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.595982e-01   9.488775e-01   9.561703e-01   9.628295e-01   2.880542e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   6.750168e-17
```

```
confusionMatrix(training$classe, predict(rfModel,training))$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9997315      1.0000000      0.2843416
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Model validation

Let's now validate them on the hold-out sample (e.g. data frame = "testing")

```
confusionMatrix(holdout$classe, predict(treeModel,holdout))$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   5.482541e-01   4.308718e-01   5.329143e-01   5.635256e-01   2.664888e-01
## AccuracyPValue  McnemarPValue
##  1.142013e-318  9.402801e-119
```

```
confusionMatrix(holdout$classe, predict(bagModel,holdout))$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9997575      0.9996934      0.9986497      0.9999939      0.2817653
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
confusionMatrix(holdout$classe, predict(boostModel,holdout))$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.587779e-01   9.478501e-01   9.522556e-01   9.646396e-01   2.861300e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   1.009525e-05
```

```
confusionMatrix(holdout$classe, predict(rfModel,holdout))$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9991059      1.0000000      0.2817653
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

So the best model is 'rfModel', which was built using Random Forest. The bagging approach also yielded a very good model. For rfModel the in sample error (1-Accuracy) 0%, and the out of sample error was also 0%. We can further see that the model is correctly predicting all 'classe' classifications by showing the full confusionMatrix for both the training and hold-out sample.

```
confusionMatrix(training$classe, predict(rfModel,training))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
confusionMatrix(holdout$classe, predict(rfModel,holdout))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1162    0    0    0    0
##          B    0  803    0    0    0
##          C    0    0  711    0    0
##          D    0    0    0  658    0
##          E    0    0    0    0  790
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2818
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2818   0.1947   0.1724   0.1596   0.1916
## Detection Rate         0.2818   0.1947   0.1724   0.1596   0.1916
## Detection Prevalence   0.2818   0.1947   0.1724   0.1596   0.1916
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Quiz answers

Finally, we can score the ‘testing’ set which is needed for the quiz

```
answers_for_quiz <- predict(rfModel, testing)
as.vector(answers_for_quiz)
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```