Prediction Assignment Writeup

A Salancy

November 11, 2017

Business Problem

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, we use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise. This is the "classe" variable in the training set. Other variables are used to predict with. This report describes how we built the model, used cross validation, what we think the expected out of sample error is, and why we made the choices we did. We also use the prediction model to predict 20 different test cases.

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://web.archive.org/web/20161224072740/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 have been very generously provided from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz4y8X69116

Load Necessary Libraries

The packages for the following libraries have already been installed.

```
library(caret)
library(randomForest)
library(e1071)
library(rpart)
```

```
library(AppliedPredictiveModeling)
library(ElemStatLearn)
library(gbm)
library(lubridate)
```

Data Acquisition

```
Train_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
Test_url  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(Train_url))
testing  <- read.csv(url(Test_url))
```

Data Pre-processing

We set a seed for reproducibility.

```
set.seed(31415)
```

Note that any operations applied to the training set will need to be applied to the testing set as well.

```
dim(training)
```

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

We're starting off with 160 variables. First, remove unnecessary columns. The first 6 columns provide user name and time stamps.

```
training <- training[,-c(1:6)]
testing  <- testing[,-c(1:6)]
```

Next, remove variables where more than half of the values are NA.

```
High_NA<- sapply(training, function(x) mean(is.na(x))) > 0.5
training <- training[, High_NA==FALSE]
testing <- testing[, High_NA==FALSE]
```

Next, remove Near Zero Variance variables.

```
NZV <- nearZeroVar(training)
training <- training[, -NZV]
testing <- testing[, -NZV]
dim(training)
```

```
## [1] 19622    54
```

We're now down to 54 variables.

Model Building

We now set aside data for validation, keeping 75% of the data for training. Each model below is built with the *mytrain* training data subset and cross-validated on the *myvalidate* subset.

```
trainIndex = createDataPartition(training$classe,p=0.75,list=FALSE)
mytrain = training[trainIndex,]
myvalidate = training[-trainIndex,]
```

Random Forest

Our first model will be a random forest model. It will include 3-fold cross-validation. Note that this takes quite a while to run.

```
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRF<- train(classe ~ ., method="rf", data=mytrain,trControl=controlRF)
modFitRF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4183    1    0    0    1 0.0004778973
## B    7 2838    2    1    0 0.0035112360
## C    0    4 2563    0    0 0.0015582392
## D    0    0    7 2405    0 0.0029021559
## E    0    1    0    5 2700 0.0022172949
```

We test the model on the validation set to understand its level of accuracy.

```
predRF<-predict(modFitRF, myvalidate)
cmRF<-confusionMatrix(predRF, myvalidate$classe)
cmRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  947    2    0    0
##          C    0    1  853    0    0
##          D    0    0    0  803    3
##          E    0    0    0    1  898
##
## Overall Statistics
##
##                Accuracy : 0.9984
##                  95% CI : (0.9968, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9979   0.9977   0.9988   0.9967
## Specificity            0.9997   0.9995   0.9998   0.9993   0.9998
```

```
## Pos Pred Value         0.9993   0.9979   0.9988   0.9963   0.9989
## Neg Pred Value         1.0000   0.9995   0.9995   0.9998   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1931   0.1739   0.1637   0.1831
## Detection Prevalence   0.2847   0.1935   0.1741   0.1644   0.1833
## Balanced Accuracy      0.9999   0.9987   0.9987   0.9990   0.9982
```

The random forest model produces an accuracy on this validation set of over 99.8%. This is very good, but we'll check two other methods: boosting and bootstrap aggregating, or "bagging".

Boosting

Note that this takes quite a while to run.

```
modFitBoo<- train(classe ~ ., method="gbm", data=mytrain,verbose=FALSE)
```

```
##
## Attaching package: 'plyr'
```

```
## The following object is masked from 'package:lubridate':
##
##     here
```

```
## The following object is masked from 'package:ElemStatLearn':
##
##     ozone
```

```
modFitBoo$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 42 had non-zero influence.
```

We test the model on the validation set to understand its level of accuracy.

```
predBoo<-predict(modFitBoo, myvalidate)
cmBoo<-confusionMatrix(predBoo, myvalidate$classe)
cmBoo
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1391    8    0    1    1
##          B    4  929    6    2    6
##          C    0   12  843    9    5
##          D    0    0    5  792    8
##          E    0    0    1    0  881
##
## Overall Statistics
##
##                Accuracy : 0.9861
##                  95% CI : (0.9825, 0.9892)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9825
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9971   0.9789   0.9860   0.9851   0.9778
## Specificity            0.9972   0.9954   0.9936   0.9968   0.9998
## Pos Pred Value         0.9929   0.9810   0.9701   0.9839   0.9989
## Neg Pred Value         0.9989   0.9949   0.9970   0.9971   0.9950
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2836   0.1894   0.1719   0.1615   0.1796
## Detection Prevalence   0.2857   0.1931   0.1772   0.1642   0.1799
## Balanced Accuracy      0.9971   0.9872   0.9898   0.9910   0.9888
```

The boosting model produces an accuracy on this validation set of 98.5%.

Bagging

Note that this takes quite a while to run.

```
modFitBag<- train(classe ~ ., method="treebag", data=mytrain)
modFitBag$finalModel
```

```
##
## Bagging classification trees with 25 bootstrap replications
```

We test the model on the validation set to understand its level of accuracy.

```
predBag<-predict(modFitBag, myvalidate)
cmBag<-confusionMatrix(predBag, myvalidate$classe)
cmBag
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    7    0    0    0
##          B    0  938    2    1    0
##          C    0    4  853    2    0
##          D    0    0    0  800    3
##          E    0    0    0    1  898
##
## Overall Statistics
##
##                Accuracy : 0.9959
##                  95% CI : (0.9937, 0.9975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9948
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9884   0.9977   0.9950   0.9967
## Specificity            0.9980   0.9992   0.9985   0.9993   0.9998
```

```
## Pos Pred Value         0.9950   0.9968   0.9930   0.9963   0.9989
## Neg Pred Value         1.0000   0.9972   0.9995   0.9990   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1913   0.1739   0.1631   0.1831
## Detection Prevalence   0.2859   0.1919   0.1752   0.1637   0.1833
## Balanced Accuracy      0.9990   0.9938   0.9981   0.9971   0.9982
```

The bagging model produces an accuracy on this validation set of 99.6%.

Checking Out-of-Sample Error

```
OSE_Bag <- 1-sum(predBag == myvalidate$classe)/length(predBag)
OSE_Bag
```

```
## [1] 0.004078303
```

The out-of-sample error is extremely low in the examined case of the bagging model.

Applying Model to Test Set

All three models produce very high accuracy. We'll use the bagging model on our test set.

```
predBag<-predict(modFitBag, testing)
predBag
```

```
##  [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
```

Conclusions

```
length(predBag)
```

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

```
table(predBag)
```

```
## predBag
## A B C D E
## 7 8 1 1 3
```

Only 35% (7 of 20) of subjects in the test case are modeled to be using proper form, as indicated by an "A" classification. The others are making the common mistakes indicated by the other classes.