Practical Machine Learning - Recognizing Qualitative Activity

Atit Doctor

February 25, 2016

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, we will use data recorded from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

The goal of this project is to predict the manner in which the participants did the exercise. This is the classe variable of the training set, which classifies the correct and incorrect outcomes into A, B, C, D, and E categories. This report describes how the model for the project was built, its cross validation, expected out of sample error calculation, and the choices made. It was used successfully to accurately predict all 20 different test cases on the Coursera website.

Data Loading and Cleaning

```
# Set working directory
setwd("/Users/adoctor/Documents/PracticalMachineLearning_PredictionAssignment/")

# Read data
pmlTrain<-read.csv("pml-training.csv", header=T, na.strings=c("NA", "#DIV/0!"))
pmlTest<-read.csv("pml-testing.csv", header=T, na.string=c("NA", "#DIV/0!"))
```

Training data was partitioned and preprocessed using the code described below. In brief, all variables with at least one "NA" were excluded from the analysis. Variables related to time and user information were excluded for a total of 51 variables and 19622 class measurements. Same variables were mainteined in the test data set (Validation dataset) to be used for predicting the 20 test cases provided.

```
## NA exclusion for all available variables
noNApmlTrain<-pmlTrain[, apply(pmlTrain, 2, function(x) !any(is.na(x)))]
dim(noNApmlTrain)
```

```
## [1] 19622    60
## variables with user information, time and undefined
cleanpmlTrain<-noNApmlTrain[,-c(1:8)]
dim(cleanpmlTrain)
```

```
## [1] 19622    52
```

```
## 20 test cases provided clean info - Validation data set
cleanpmltest<-pmlTest[,names(cleanpmlTrain[,-52])]
dim(cleanpmltest)
```

```
## [1] 20 51
```

Data Partitioning and Prediction Process

The cleaned downloaded data set was subset in order to generate a test set independent from the 20 cases provided set. Partitioning was performed to obtain a 75% training set and a 25% test set.

```
#data cleaning
library(caret)
```

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

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

```
inTrain<-createDataPartition(y=cleanpmlTrain$classe, p=0.75,list=F)
training<-cleanpmlTrain[inTrain,]
test<-cleanpmlTrain[-inTrain,]
#Training and test set dimensions
dim(training)
```

```
## [1] 14718    52
```

```
dim(test)
```

```
## [1] 4904   52
```

Results

Random forest trees were generated for the training dataset using cross-validation. Then the generated algorithm was examnined under the partitioned training set to examine the accuracy and estimated error of prediction. By using 51 predictors for five classes using cross-validation at a 5-fold an accuracy of

99.2% with a 95% CI [0.989-0.994] was achieved accompanied by a Kappa value of 0.99.

```
library(caret)
set.seed(13333)
fitControl2<-trainControl(method="cv", number=5, allowParallel=T, verbose=T)
rffit<-train(classe~.,data=training, method="rf", trControl=fitControl2, verbose=F)
```

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

```
## 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
## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry=26
## - Fold1: mtry=26
## + Fold1: mtry=51
## - Fold1: mtry=51
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry=26
## - Fold2: mtry=26
## + Fold2: mtry=51
## - Fold2: mtry=51
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry=26
## - Fold3: mtry=26
## + Fold3: mtry=51
## - Fold3: mtry=51
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry=26
## - Fold4: mtry=26
## + Fold4: mtry=51
## - Fold4: mtry=51
## + Fold5: mtry= 2
## - Fold5: mtry= 2
## + Fold5: mtry=26
## - Fold5: mtry=26
## + Fold5: mtry=51
## - Fold5: mtry=51
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 26 on full training set
```

```
predrf<-predict(rffit, newdata=test)
confusionMatrix(predrf, test$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1394   11    0    0    0
##          B    1  937    2    0    0
##          C    0    1  852    6    0
##          D    0    0    1  798    0
##          E    0    0    0    0  901
## 
## Overall Statistics
## 
##                Accuracy : 0.9955          
##                  95% CI : (0.9932, 0.9972)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.9943          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9874   0.9965   0.9925   1.0000
## Specificity            0.9969   0.9992   0.9983   0.9998   1.0000
## Pos Pred Value         0.9922   0.9968   0.9919   0.9987   1.0000
## Neg Pred Value         0.9997   0.9970   0.9993   0.9985   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1911   0.1737   0.1627   0.1837
## Detection Prevalence   0.2865   0.1917   0.1752   0.1629   0.1837
## Balanced Accuracy      0.9981   0.9933   0.9974   0.9961   1.0000
```

```
pred20<-predict(rffit, newdata=cleanpmltest)
# Output for the prediction of the 20 cases provided
pred20
```

```
##  [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
```

A boosting algorithm was also run to confirm and be able to compare predictions. Data is not shown but the boosting approach presented less accuracy (96%) (Data not shown). However, when the predictions for the 20 test cases were compared match was same for both ran algorimths.

```
fitControl2<-trainControl(method="cv", number=5, allowParallel=T, verbose=T)
gmbfit<-train(classe~.,data=training, method="gbm", trControl=fitControl2, verbose=F)
```

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

```
##
## 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
## + Fold1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## Aggregating results
## Selecting tuning parameters
## Fitting n.trees = 150, interaction.depth = 3, shrinkage = 0.1, n.minobsinnode = 10 on
full training set
```

```
gmbfit$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 51 predictors of which 44 had non-zero influence.
```

```
class(gmbfit)
```

```
## [1] "train"         "train.formula"
```

```
predgmb<-predict(gmbfit, newdata=test)
confusionMatrix(predgmb, test$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1361   36    0    2    0
##          B   20  877   25    1    7
##          C    9   29  819   27    7
##          D    2    2    9  764    9
##          E    3    5    2   10  878
## 
## Overall Statistics
## 
##                Accuracy : 0.9582          
##                  95% CI : (0.9522, 0.9636)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.9471          
##  Mcnemar's Test P-Value : 0.001087        
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9756   0.9241   0.9579   0.9502   0.9745
## Specificity            0.9892   0.9866   0.9822   0.9946   0.9950
## Pos Pred Value         0.9728   0.9430   0.9192   0.9720   0.9777
## Neg Pred Value         0.9903   0.9819   0.9910   0.9903   0.9943
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2775   0.1788   0.1670   0.1558   0.1790
## Detection Prevalence   0.2853   0.1896   0.1817   0.1603   0.1831
## Balanced Accuracy      0.9824   0.9554   0.9701   0.9724   0.9847
```

```
predtrain<-predict(gmbfit, newdata=training)
confusionMatrix(predtrain, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4139   65    0    0    4
##          B   31 2723   50    3   17
##          C    9   52 2487   72   21
##          D    5    3   26 2321   23
##          E    1    5    4   16 2641
##
## Overall Statistics
##
##                Accuracy : 0.9723
##                  95% CI : (0.9696, 0.9749)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.965
##  Mcnemar's Test P-Value : 7.44e-11
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9890   0.9561   0.9688   0.9623   0.9760
## Specificity            0.9934   0.9915   0.9873   0.9954   0.9978
## Pos Pred Value         0.9836   0.9642   0.9417   0.9760   0.9903
## Neg Pred Value         0.9956   0.9895   0.9934   0.9926   0.9946
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2812   0.1850   0.1690   0.1577   0.1794
## Detection Prevalence   0.2859   0.1919   0.1794   0.1616   0.1812
## Balanced Accuracy      0.9912   0.9738   0.9781   0.9788   0.9869
```

```
predtrain<-predict(gmbfit, newdata=training)
confusionMatrix(predtrain, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4139   65    0    0    4
##          B   31 2723   50    3   17
##          C    9   52 2487   72   21
##          D    5    3   26 2321   23
##          E    1    5    4   16 2641
##
## Overall Statistics
##
##                Accuracy : 0.9723
##                  95% CI : (0.9696, 0.9749)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.965
##  Mcnemar's Test P-Value : 7.44e-11
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9890   0.9561   0.9688   0.9623   0.9760
## Specificity            0.9934   0.9915   0.9873   0.9954   0.9978
## Pos Pred Value         0.9836   0.9642   0.9417   0.9760   0.9903
## Neg Pred Value         0.9956   0.9895   0.9934   0.9926   0.9946
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2812   0.1850   0.1690   0.1577   0.1794
## Detection Prevalence   0.2859   0.1919   0.1794   0.1616   0.1812
## Balanced Accuracy      0.9912   0.9738   0.9781   0.9788   0.9869
```

Conclusion

Once, the predictions were obtained for the 20 test cases provided, the below shown script was used to obtain single text files to be uploaded to the courses web site to comply with the submission assigment. 20 out of 20 hits also confirmed the accuracy of the obtained models.