Practical Machine Learning - Assignment Writeup

Lance Teo

Sunday, November 22, 2015

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 Cleaning and Processing

Downloading the Data

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = "./pml-
download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "./pml-t
```

Reading the Data

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

Cleaning the Data

- Removing NA

```
train.nona<-train[, apply(train, 2, function(x) !any(is.na(x)))]
dim(train.nona)
```

```
## [1] 19622    60
```

- Removing variables not related to dependent variables

```
remove = c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2', 'cvtd_timestamp', 'new_win
train.nore <- train.nona[, -which(names(train.nona) %in% remove)]
dim(train.nore)
```

```
## [1] 19622    53
```

Data Partitioning and Cross Validation

Load Required Library

```r
library(caret)
```

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

Data Partitioning (75% Training, 25% Testing)

```r
trainIndex <- createDataPartition(y=train.nore$classe, p=0.75, list=FALSE)
training <- train.nore[trainIndex,]
testing <- train.nore[-trainIndex,]
```

Dimension of Partitions

```r
dim(training)
```

```
## [1] 14718    53
```

```r
dim(testing)
```

```
## [1] 4904   53
```

Results

Generating Random Forest Trees for Training Set using Cross Validation

```r
set.seed(4444)
control<-trainControl(method="cv", number=5, allowParallel=T, verbose=T)
rf<-train(classe~.,data=training, method="rf", trControl=control, verbose=F)
```

```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry=27
## - Fold1: mtry=27
## + Fold1: mtry=52
## - Fold1: mtry=52
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry=27
## - Fold2: mtry=27
## + Fold2: mtry=52
## - Fold2: mtry=52
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry=27
## - Fold3: mtry=27
## + Fold3: mtry=52
## - Fold3: mtry=52
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry=27
## - Fold4: mtry=27
## + Fold4: mtry=52
## - Fold4: mtry=52
## + Fold5: mtry= 2
## - Fold5: mtry= 2
## + Fold5: mtry=27
## - Fold5: mtry=27
## + Fold5: mtry=52
## - Fold5: mtry=52
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 27 on full training set
```

Examine the accuracy and estimated error of predicrion

```r
predrf<-predict(rf, newdata=testing)
confusionMatrix(predrf, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392   17    0    0    0
##          B    1  929    4    0    0
##          C    1    2  847   10    0
##          D    0    1    4  792    1
##          E    1    0    0    2  900
##
## Overall Statistics
##
##                Accuracy : 0.991
##                  95% CI : (0.988, 0.9935)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9886
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9789   0.9906   0.9851   0.9989
## Specificity            0.9952   0.9987   0.9968   0.9985   0.9993
## Pos Pred Value         0.9879   0.9946   0.9849   0.9925   0.9967
## Neg Pred Value         0.9991   0.9950   0.9980   0.9971   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2838   0.1894   0.1727   0.1615   0.1835
## Detection Prevalence   0.2873   0.1905   0.1754   0.1627   0.1841
## Balanced Accuracy      0.9965   0.9888   0.9937   0.9918   0.9991
```

Conclusion

An accuracy of 99.39% with a 95% CI [0.9915-0.9957] was achieved accompanied by a Kappa value of 0.9923.

Prediction of the Testing Set given

```r
pred<-predict(rf, newdata=test)
pred
```

```
##  [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
```