Practical Machine Learning - Peer Assessment

John Allen

Thursday, March 24, 2016

Executive Summary

For the Practical Machine Learning Peer Assessment we will build a prediction model to predict the manner in which a subject has excersised from the data collected via their personal activity device.

By using a random forests model and removing the columns that have less than 60% of the data entered, a predictive model accuracy over the validation dataset of 99.9% is acheived.

Load the libraries and Set Seed

```
setwd("C:/Users/Jono/Desktop/Coursera/Practical_Machine_Learning")
library(rpart)
library(caret)
library(randomForest)
library(rpart.plot)
library(corrplot)
```

Load all libraries used, and set the working directory

```
training <- read.csv("./Data/pml-training-1.csv", header=TRUE)
testing <- read.csv("./Data/pml-testing.csv", header=TRUE)
dim(training)
dim(testing)
```

Load the training and testing datasets

DATA PROCESSING

```
sum(complete.cases(training))
```

First we will clean the data of obeservations with missing values, observations with near zero variance and the variables that are not needed for the exercise. I will then partition the data in to two datasets.

```
## [1] 406
```

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
dim(training)
```

Removing incomplete columns

```
## [1] 19622    93
```

```
training <- training[, -(1:5)]
```

Removing unneeded columns, which happens to be the first 5 columns

```
nzv <- nearZeroVar(training)
training <- training[, -nzv]
dim(training)
```

Removing observations with near zero variance

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

```
set.seed(100)
inTrain = createDataPartition(training$classe, p=0.60, list=FALSE)
training1 = training[inTrain,]
validating1 = training[-inTrain,]
```

Here I will seperate the data (training1) into a smaller set and validation set (training2). This is done to be able to estimate the out-of-sample error rate.

MODEL BUILD

```
FrstModel <- randomForest(classe~.,data=training1)
print(FrstModel)
```

First we will fit a random forest mode, check the importance and then check the performance on the validation1 set by running it through a confusion matrix.

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training1)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.41%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B    8 2269    2    0    0 0.0043878894
## C    0   13 2040    1    0 0.0068159688
## D    0    0   16 1913    1 0.0088082902
## E    0    0    0    6 2159 0.0027713626
```

```
#check the importance
importance(FrstModel)
```

```
##                      MeanDecreaseGini
## num_window                  801.20698
## roll_belt                   683.12828
## pitch_belt                  382.10212
## yaw_belt                    465.28799
## total_accel_belt            121.93911
## gyros_belt_x                 54.98491
## gyros_belt_y                 60.51736
## gyros_belt_z                151.76583
## accel_belt_x                 73.48702
## accel_belt_y                 80.63817
## accel_belt_z                226.03406
## magnet_belt_x               145.76633
## magnet_belt_y               232.75899
## magnet_belt_z               217.35234
## roll_arm                    181.70451
## pitch_arm                    95.52567
## yaw_arm                     121.29874
## total_accel_arm              56.50341
## gyros_arm_x                  63.92826
## gyros_arm_y                  67.45323
## gyros_arm_z                  30.75469
## accel_arm_x                 129.31234
## accel_arm_y                  80.07395
## accel_arm_z                  66.78007
## magnet_arm_x                137.37949
## magnet_arm_y                129.31263
## magnet_arm_z                102.58054
## roll_dumbbell               242.07489
## pitch_dumbbell               99.73416
## yaw_dumbbell                146.50947
## total_accel_dumbbell        157.68255
## gyros_dumbbell_x             69.38701
## gyros_dumbbell_y            127.88450
## gyros_dumbbell_z             43.57506
```

```
## accel_dumbbell_x            147.29202
## accel_dumbbell_y            228.75821
## accel_dumbbell_z            188.83927
## magnet_dumbbell_x           271.13144
## magnet_dumbbell_y           386.67687
## magnet_dumbbell_z           420.49106
## roll_forearm                305.55167
## pitch_forearm               457.02888
## yaw_forearm                  90.29339
## total_accel_forearm          59.26231
## gyros_forearm_x              40.79132
## gyros_forearm_y              65.55596
## gyros_forearm_z              50.50348
## accel_forearm_x             177.72120
## accel_forearm_y              73.01044
## accel_forearm_z             137.52532
## magnet_forearm_x            111.21488
## magnet_forearm_y            110.94804
## magnet_forearm_z            139.85184
```

```
predictRf <- predict(FrstModel, validating1)
confusionMatrix(validating1$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    4 1511    3    0    0
##          C    0    6 1360    2    0
##          D    0    0    6 1280    0
##          E    0    0    0    6 1436
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.995, 0.9977)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9956
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9960   0.9934   0.9938   1.0000
## Specificity            1.0000   0.9989   0.9988   0.9991   0.9991
## Pos Pred Value         1.0000   0.9954   0.9942   0.9953   0.9958
## Neg Pred Value         0.9993   0.9991   0.9986   0.9988   1.0000
## Prevalence             0.2850   0.1933   0.1745   0.1642   0.1830
## Detection Rate         0.2845   0.1926   0.1733   0.1631   0.1830
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9991   0.9975   0.9961   0.9964   0.9995
```

```
accuracy<-c(as.numeric(predict(FrstModel,newdata=validating1[,-ncol(validating1)])==validating1$classe)
accuracy<-sum(accuracy)*100/nrow(validating1)
accuracy
```

```
## [1] 99.65588
```

Here it is shown the accuracy of the model applied to the validation set (validating1) comes in at 99.65%

```
FinalTest <- predict(FrstModel, testing)
FinalTest
```

The only step left is to apply the model to the original testing set downloaded aquired from the data source.

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```