MachineLearningAssignment

Ravi Keron

Sunday, January 31, 2016

Introduction

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.

Data

The training data for this work is available at https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The testing data for this work is available at https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Goal

The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. A report is to be created describing how the model is built, how cross validation is used, what the expected out of sample error is, and why the choices were made so.

Loading the data into training and testing

```
library(caret)
```

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

```
library(kernlab)
set.seed(32323)
setwd("E:/DataScience/Practical Machine Learning/assignment")
training <- read.csv("pml-training.csv", na.strings=c("NA", "#DIV/0!", ""))
testing<-read.csv("pml-testing.csv",  na.strings=c("NA", "#DIV/0!", ""))
```

Data Observation and cleanup

```
str(training, list.len = 10)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2
2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 13230842
32 1323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440
390 484323 484434 ...
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9
9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45
...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17
...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.
4 -94.4 ...
##   [list output truncated]
```

Looks like the data has plenty of NAs.Also to decide the predictors, a check and removal of Near Zero variance elements has to be done.A basic cleanup has to be done.

Remove columns that are mostly have NA’s

The first 6 columns does not seem to contribute in the prediction as they are just timestamps and names. These can be removed alongwith mostly NA columns

```
training <- training[c(-1, -2, -3, -4, -5, -6)]
testing<-testing[c(-1, -2, -3, -4, -5, -6)]
NA_data <- apply(!is.na(training), 2, sum) > 19621
training<-training[, NA_data]
testing <- testing[, NA_data]
```

Check for near zero vars

```
nzvtrain <- nearZeroVar(training, saveMetrics=TRUE)
training <- training[, nzvtrain$nzv==FALSE]

nzvtest <- nearZeroVar(testing, saveMetrics=TRUE)
testing <- testing[, nzvtest$nzv==FALSE]
```

Now the next step is to build the model. Break the training set into two parts 60% for training and 40% for testing

```
## [1] 11776    54
```

```
## [1] 7846   54
```

Prediction Model

Let us apply the Random Forest model to see what accuracy that the model provide. Based on the output other models can be tried if the accuracy is not beyond 90% We use the trainingset1 to train the model and do the fit model with testingset1

Fitting the model with Random Forest

```
set.seed(12345)
library(doParallel)
```

```
## Loading required package: foreach
## Loading required package: iterators
## Loading required package: parallel
```

```
registerDoParallel(cores=2)
modelfit1 <- train(classe ~., data=trainingset1, method = "rf")
```

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

```
modelfit1
```

```
## Random Forest
##
## 11776 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9897181  0.9869884  0.001640681  0.002072457
##   27    0.9946876  0.9932781  0.001117536  0.001413230
##   53    0.9913947  0.9891098  0.002214046  0.002810600
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Check the model accuracy using training set and then the testing set created from the training set

```
predictTrain <- predict(modelfit1, trainingset1)
modelfit1$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: 0.32%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3346    1    0    0    1 0.0005973716
## B   11 2264    4    0    0 0.0065818341
## C    0    7 2046    1    0 0.0038948393
## D    0    0    9 1921    0 0.0046632124
## E    0    0    0    4 2161 0.0018475751
```

```
confusionMatrix(predictTrain, trainingset1$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## 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
```

```
predtest <- predict(modelfit1, testingset1)
confusionMatrix(predtest, testingset1$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    5    0    0    0
##          B    0 1512    0    0    0
##          C    0    0 1368    0    0
##          D    0    1    0 1286    2
##          E    0    0    0    0 1440
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.998, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9960   1.0000   1.0000   0.9986
## Specificity            0.9991   1.0000   1.0000   0.9995   1.0000
## Pos Pred Value         0.9978   1.0000   1.0000   0.9977   1.0000
## Neg Pred Value         1.0000   0.9991   1.0000   1.0000   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1927   0.1744   0.1639   0.1835
## Detection Prevalence   0.2851   0.1927   0.1744   0.1643   0.1835
## Balanced Accuracy      0.9996   0.9980   1.0000   0.9998   0.9993
```

Looking at the testing part of the training set the accuracy of the model is at 99% which is very good and with a very small out of sample error and with a CI between 99.5% and 99.77%

check the important predictors

```
varImp(modelfit1)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 53)
##
##                      Overall
## num_window           100.000
## roll_belt             66.174
## pitch_forearm         39.641
## yaw_belt              32.689
## magnet_dumbbell_z     31.560
## pitch_belt            28.864
## magnet_dumbbell_y     27.253
## roll_forearm          20.560
## accel_dumbbell_y      13.386
## magnet_dumbbell_x     12.062
## roll_dumbbell         11.446
## accel_forearm_x       10.799
## accel_belt_z           9.378
## total_accel_dumbbell   8.740
## accel_dumbbell_z       8.297
## magnet_belt_y          7.330
## magnet_forearm_z       6.799
## magnet_belt_z          6.409
## magnet_belt_x          5.698
## gyros_belt_z           5.339
```

```
modelfit1$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: 0.32%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3346    1    0    0    1 0.0005973716
## B   11 2264    4    0    0 0.0065818341
## C    0    7 2046    1    0 0.0038948393
## D    0    0    9 1921    0 0.0046632124
## E    0    0    0    4 2161 0.0018475751
```

Apply the prediction model on the testing set

The model is applied on the test data

```
apply_predtest <- predict(modelfit1, testing)
apply_predtest
```

```
##  [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
```

Finally produce the files for submission

```
pml_write_files=function(x){
n=length(x)
for(i in 1:n){
filename=paste0("problem_id_",i,".txt")
write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
}
}
pml_write_files(apply_predtest)
```

20 test cases were predicted using the model and it went 100% accurate