Practical Machine Learning - Peer Assessment

Nidhi Mavani

Saturday, December 20, 2014

An Analysis of the Weight Lifting Exercises Dataset

This document presents the results of the Practical Machine Learning Peer Assessments in a report using a single R markdown document that can be processed by knitr and be transformed into an HTML file.

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 Â- 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 data set, the participants 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).
In this project, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants toto predict the manner in which praticipants did the exercise.
Since we have a data set with too many columns and we need make a class prediction, therefore decided to implement a random forests model, that's no need cross-validation or a separate test set to get an unbiased estimate of the test set error.

The dependent variable or response is the "classe" variable in the training set.

Data

Getting And Cleaning of Data

Load the data.

```
training <- read.csv("pml-training.csv", header=T, sep=",", na.strings=c("NA",""))
testing<-read.csv("pml-testing.csv", header=T,sep=",", na.strings=c("NA",""))
dim(training)
```

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

```
dim(testing)
```

```
## [1]  20 160
```

Data Exploration

Firstly Load the Libraries and Set Seed for results to be Reproducible.

```
library(caret)
library(randomForest)
set.seed(45)
```

The following approaches for used for reducing the number of predictors.

- Remove variables that have too many NA values.
  As there are many variables that are of value NA we remove those columns that have 70% of NAs,inorder to find out which Columns satisfy the criteria

```
##this will return the total number of Non NAs values in each Column
ColSums<-colSums(!is.na(training[,-ncol(training)]))
head(ColSums)
```

```
##                    X            user_name raw_timestamp_part_1
##                19622                19622                19622
## raw_timestamp_part_2       cvtd_timestamp           new_window
##                19622                19622                19622
```

```
## this gives the number of valid columns
sum(colSums(!is.na(training[,-ncol(training)]))>=0.7*nrow(training))
```

```
## [1] 59
```

Subset the Data by considering only valid columns

```
validCol<-colSums(!is.na(training[,-ncol(training)]))>=0.7*nrow(training)
trainingWithlessNA<-training[,validCol]
dim(trainingWithlessNA)
```

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

- Remove columns having near zero variance

```
nzv <- nearZeroVar(trainingWithlessNA, saveMetrics = TRUE) ##this creates a matrix
head(nzv)
```

```
##                      freqRatio percentUnique zeroVar   nzv
## X                     1.000000  100.00000000   FALSE FALSE
## user_name             1.100679    0.03057792   FALSE FALSE
## raw_timestamp_part_1  1.000000    4.26562022   FALSE FALSE
## raw_timestamp_part_2  1.000000   85.53154622   FALSE FALSE
## cvtd_timestamp        1.000668    0.10192641   FALSE FALSE
## new_window           47.330049    0.01019264   FALSE  TRUE
```

```
#this will return only those col numbers
nzv <- nearZeroVar(trainingWithlessNA)
trainingWithNozeroVar <- trainingWithlessNA[, -nzv]
dim(trainingWithNozeroVar)
```

```
## [1] 19622    59
```

- Remove the ID Variable

```
trainingWithNozeroVarandID<-trainingWithNozeroVar[,-1]
dim(trainingWithNozeroVarandID)
```

```
## [1] 19622    58
```

- Remove those variable having high correlation

```
##number of the columns with numeric values
sum(sapply(trainingWithNozeroVarandID, is.numeric))
```

```
## [1] 55
```

```
##VAriables with high correraltion
corrMatrix <-
  cor(na.omit(trainingWithNozeroVarandID[sapply(trainingWithNozeroVarandID, is.numeric)]))
dim(corrMatrix)
```

```
## [1] 55 55
```

```
##A cut-off of 90% is kept
highCorr<-findCorrelation(corrMatrix, cutoff = .90, verbose = T)
## Final Training set
Final_training<-trainingWithNozeroVarandID[,-highCorr]
```

```
dim(Final_training)
```

```
## [1] 19622    51
```

The same is to be done with testing Data

```
testingWithLessNA<-testing[,validCol]
testingWithNoZeroVar<-testingWithLessNA[,-nzv]
testingWithNoZeroVarAndID<-testingWithNoZeroVar[,-1]
Final_testing<-testingWithNoZeroVarAndID[,-highCorr]
dim(Final_testing)
```

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

Analysis

Partitioning the training Data

```
set.seed(45)
inTrain<-createDataPartition(Final_training$classe,p=0.75,list=F)
modelTrain<-Final_training[inTrain,]
modeltest<-Final_training[-inTrain,]
```

Model Fitting

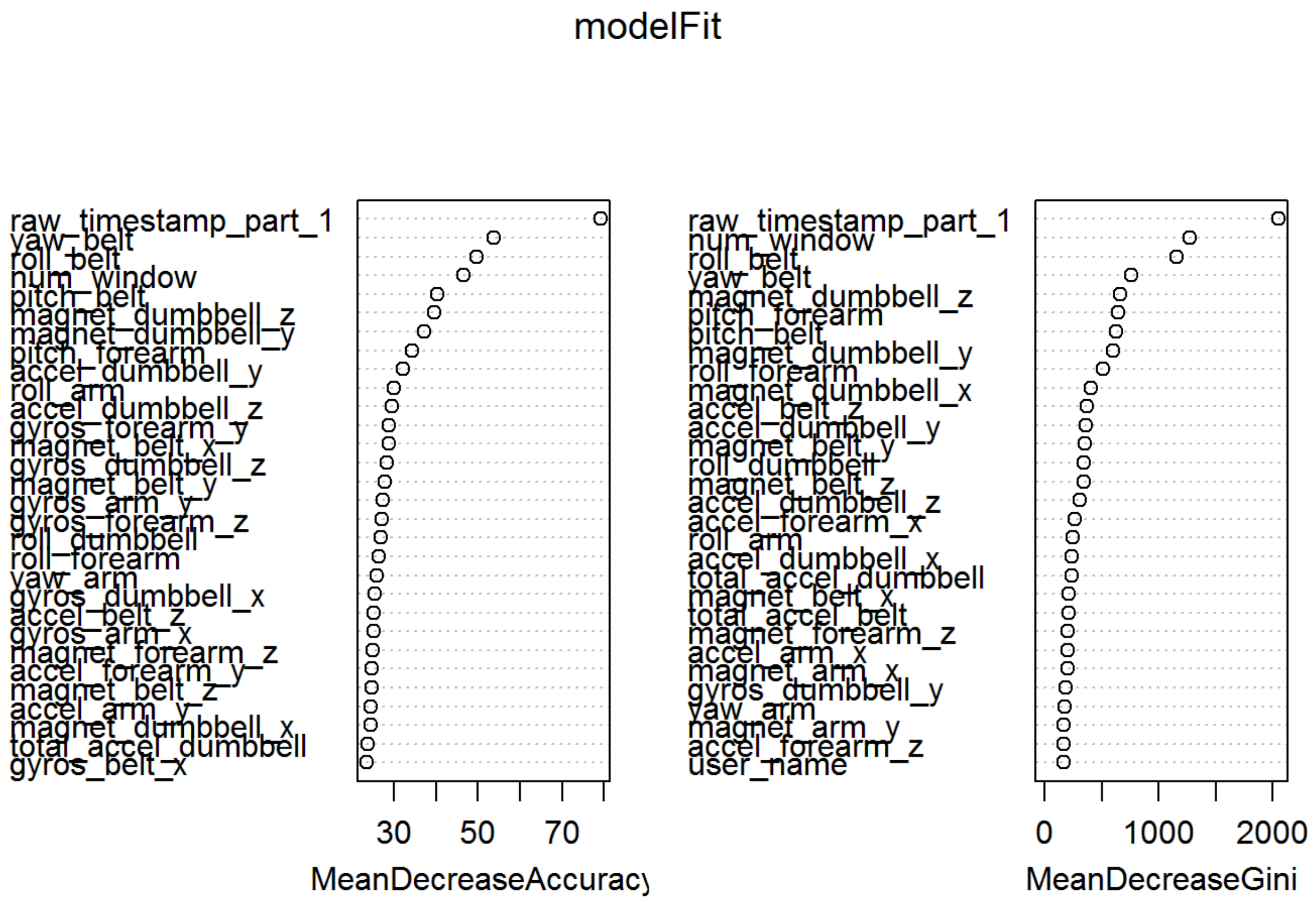
Random forests build lots of bushy trees, and then average them to reduce the variance.
In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, during the execution. So, we proced with the training the model (Random Forest) with the training data set.

```
set.seed(45)
require(randomForest)
modelFit<-randomForest(classe~.,data=Final_training,importance=TRUE)
modelFit
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = Final_training, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.05%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5580    0    0    0    0 0.0000000000
## B    2 3795    0    0    0 0.0005267316
## C    0    5 3417    0    0 0.0014611338
## D    0    0    1 3214    1 0.0006218905
## E    0    0    0    0 3607 0.0000000000
```

No. of variables tried at each split: 7. It means every time we only randomly use 7 predictors to grow the tree.

```
varImpPlot(modelFit)
```



Out-of Sample Accuracy

Now lets evaluate this tree on the testing data.
Accuracy is 1 (100%)

```
confusionMatrix(predict(modelFit,newdata=modeltest[,-ncol(modeltest)]),
                modeltest$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    0    0    0
##          C    0    0  855    0    0
##          D    0    0    0  804    0
##          E    0    0    0    0  901
## 
## Overall Statistics
## 
##                Accuracy : 1
##                  95% CI : (0.9992, 1)
##     No Information Rate : 0.2845
##     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             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Prediction

Now lets Predict for the Final_testing Data

```
predictions <- predict(modelFit,newdata=Final_testing)
predictions
```

```
##  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
```

Those answers were submitted as a part of Course and are Correct