Practical Machine Learning Project

PQ

31 January 2016

1.Executive Summary

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. In this project, we will be using the data collected 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.The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

The goal is to predict the manner in which they did the exercise from a set of 20 records (validate_DF). This is the "classe" variable (values are A, B, C, D, E) in the training set.

2.Loading the Libraries and Getting the Data

```
#Load libraries
library(caret)

## Warning: package 'caret' was built under R version 3.2.3

## Loading required package: lattice

## Loading required package: ggplot2

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.2.3

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

set.seed(123)

#Load the data required
validate_DF = read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!"))
all_data = read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!"))
```

A quick look at the data. We can see that there quite a number of variables with a lot of missing (NA) values or error values (#DIV/0)

```
summary(all_data)
```

```
summary(all_data$classe)
```

3.Cleaning the Data

Now, we will proceed to tidy up the data before creating our models

```
#Removing variables with high number of NAs
all_data_rownum <- nrow(all_data)
all_data_colnum <- length(all_data)

#Create a new data frame that will be used for the training model
all_data_clean <- all_data

#Remvoving columns/variables with high number (90%) of NA values.
#We are left with 60 columns.
all_data_clean <- all_data_clean[, colSums(is.na(all_data_clean)) <
                                   all_data_rownum * 0.9]
#remove column 1 (index number of the dataset)
all_data_clean<- all_data_clean[,c(-1)]

#Applying the same to the validation data set, using what is the column names
that is left
remaining_col <- names(all_data_clean)
#This dataset does not have the "classe" data
remaining_col <- remaining_col[remaining_col!="classe"]
validate_DF_clean <- validate_DF[remaining_col]
```

4.Partioning the data into 2 sets.

The training data for the model is 60% and testing data set is 40%

```
inTrain = createDataPartition(all_data_clean$classe, p = 0.60)[[1]]
training = all_data_clean[ inTrain,]
testing = all_data_clean[-inTrain,]
```

5.Model Building

The random forest method is used in this model

```
Model1Control <- trainControl(method="cv", number=3, verboseIter=F)
Model1_RF <- train(classe ~ ., data=training, method="rf",
trControl=Model1Control)

Model1_RF$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: 41
##
##         OOB estimate of  error rate: 0.12%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3348    0    0    0    0 0.0000000000
## B    2 2277    0    0    0 0.0008775779
## C    0    2 2049    3    0 0.0024342746
## D    0    0    4 1925    1 0.0025906736
## E    0    0    0    2 2163 0.0009237875
```

Let's now fit the model with the testing data set, as to ensure that we are not overfitting the model based on the training set.

```r
pred_Model1_RF <- predict(Model1_RF, newdata=testing)
```

6.Accuracy of the Mode

The accuracy of the model is 99.94%, which is a very good number. In this case, we will proceed with using this model to predict the "classe" values for the 20 records in the "validate_DF_clean" data set.

```r
confusionMatrix(testing$classe,pred_Model1_RF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    0 1518    0    0    0
##          C    0    2 1365    1    0
##          D    0    0    0 1286    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9996
##                  95% CI : (0.9989, 0.9999)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9987   1.0000   0.9992   1.0000
## Specificity            1.0000   1.0000   0.9995   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   0.9978   1.0000   1.0000
## Neg Pred Value         1.0000   0.9997   1.0000   0.9998   1.0000
## Prevalence             0.2845   0.1937   0.1740   0.1640   0.1838
## Detection Rate         0.2845   0.1935   0.1740   0.1639   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   0.9993   0.9998   0.9996   1.0000
```

7.Prediction / Results

Now, getting the predictions of "classe" based on the random forest model created - whether they are "A","B","C","D" or "E".

```
pred1 <- predict(Model1_RF,validate_DF_clean,type="raw")
pred1

##  [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
```