Practical Machine Learning Project

Palash Goyal

Sunday, September 21, 2014

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. 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.

Here I have used the 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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data Sets

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Loading Data

Download the csv sheets for both training and testing datasets, and move to the respective directories.

```
setwd("C:\\Users\\palas_000\\Data Science Lectures\\Specialization\\Practical Machine Learning\\pr
oject")
training <- read.csv("pml-training.csv", header=TRUE)
testing <- read.csv("pml-testing.csv", header=TRUE)
library(caret)
library(randomForest)
library(plyr)
options(warn=-1)
```

Cleaning the Data

We have removed the unnecessary 7 columns in the starting of both the ‘training’ and ‘testing’ data frames. Some columns have NA and empty values “”, fot those columns a check of 95% has been made, and rest of the columns have been included.

```
# Removing the starting 6 columns from the training and testing datasets :
#  'X',  'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2', 'cvtd_timestamp', 'new_windo
w'
testing <- subset(testing, select=-c(1:7))
training <- subset(training, select=-c(1:7))

# Putting a Threshold value for the check on the number of occurences of 'NA' or '""' empty values
 in the datasets
threshold_val <- 0.95 * dim(training)[1]

# Here we will be removing the columns which are fully 'NA' or full empty or have more than 95 % N
A or empty values, and include rest of the columns
include_cols <- !apply(training, 2, function(y) sum(is.na(y)) > threshold_val || sum(y=="") > thre
shold_val)
training <- training[, include_cols]
```

Further Cleaning of Data

Here we are further cleaning the data by removing the predictor variables which are showing very less value of variance. In addition to this, any 2 variables which are highly correlated will not be giving any significant advantage to our prediction model. So, we have created a correlation matrix and have removed few of the highly correlated predictor variables by putting a check of the cutoff value equal to 0.95.

```
# Removing the columns which have very low variance values, and including the favourable ones
nearZvar <- nearZeroVar(training, saveMetrics = TRUE)
training <- training[ , nearZvar$nzv==FALSE]
dim(training)
```

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

```
# Making a correlation matrix to remove the columns which are highly correlated with each other
# Putting a threshold value check here as well
corr_matrix <- abs(cor(training[,-dim(training)[2]]))
# Making the default diagonal values from '1' to '0', so that these values aren't included later
diag(corr_matrix) <- 0

# Here we will be removing the columns which are highly correlated
correlated_col <- findCorrelation(corr_matrix, verbose = FALSE , cutoff = .95)
training <- training[, -c(correlated_col)]
dim(training)
```

```
## [1] 19622    49
```

```
# This removed 4 columns out of the dataframe which were highly correlated
```

Modeling the data

We have applied the Random Forest Model and it has shown significant amount of accuracy in prediction. Now we are partitioning the 'training' dataset into two parts : train1A and train2A. We have created our prediction model on train1A, and then tested it on train2A.

```
set.seed(32233)

# Dividing the training dataset in 2 parts with p=0.7
inTrain = createDataPartition(training$classe, p = 0.7, list=FALSE)
train1A <- training[inTrain,]
train2A <- training[-inTrain,]

# While creating the model, we have to specify 'importance=TRUE' for further ease in Variable-Impo
rtance plot.
# randForMod <- randomForest(classe~., data=train1A)
randomForMod <- randomForest(classe~., data=train1A, importance=TRUE)
randomForMod
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = train1A, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 0.54%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3903    1    1    0    1    0.000768
## B   12 2642    4    0    0    0.006020
## C    0   19 2374    3    0    0.009182
## D    0    0   25 2225    2    0.011989
## E    0    0    3    3 2519    0.002376
```

```
# This takes a lot of time, and gives accuracy around 96%
# ranFMod <- train(classe~., data=train2A, method="rf")


# Testing the above model on train2A  dataset of the training dataset
train2A_pred <- predict(randomForMod, newdata=train2A)
# Showing the Confusion Matrix here :
confusionMatrix(train2A_pred, train2A$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1135    4    0    0
##          C    0    0 1022    6    2
##          D    0    0    0  957    4
##          E    0    0    0    1 1076
##
## Overall Statistics
##
##                Accuracy : 0.996
##                  95% CI : (0.995, 0.998)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    0.996    0.996    0.993    0.994
## Specificity             0.999    0.999    0.998    0.999    1.000
## Pos Pred Value          0.998    0.996    0.992    0.996    0.999
## Neg Pred Value          1.000    0.999    0.999    0.999    0.999
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.193    0.174    0.163    0.183
## Detection Prevalence    0.285    0.194    0.175    0.163    0.183
## Balanced Accuracy       1.000    0.998    0.997    0.996    0.997
```

The Confusion Matrix achieved 99.59% accuracy. Here, the Out-Of-Sample Error Rate observed is 0.41%, and the OOB (Out-Of-Bag) Error Rate is 0.63%.

Checking Out-of-Sample error

On the basis of the Confusion Matrix calculated above, showing the the out-of-sample error value:

```
confM <- confusionMatrix(train2A_pred, train2A$classe)
sum(diag(confM$table))/sum(confM$table)
```

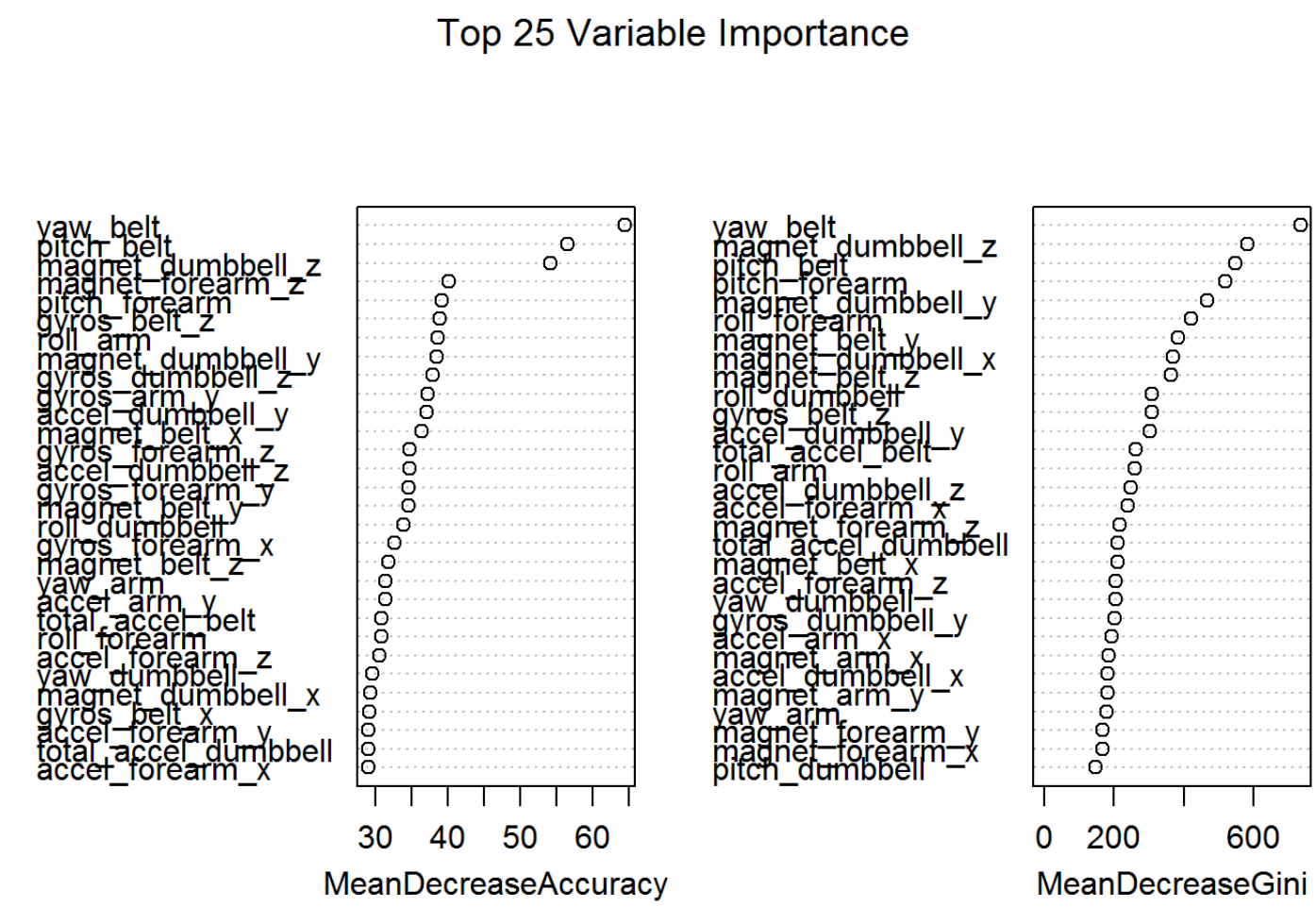
```
## [1] 0.9964
```

Variable-Importance Plot

Calculating the important predictors of the randomForest model via ‘varImp()’ function.

```
# Performing the variable importance evaluation function using the above randomForest model:
# Using 'scale=FALSE' to avoid the automatic normalization done by the varImp() function
# rfImp <- varImp(randForMod, scale=FALSE)
randomfImp <- varImp(randomForMod, scale=FALSE)
```

```
# This gives the plot for the top 25 important variables
varImpPlot(randomForMod, top=25, main="Top 25 Variable Importance")
```



Model on ‘testing’ dataset

These are the solutions for the 20 cases which have been submitted as the answers.

```
# Testing the model on the 'testing' dataset
testing_pred <- predict(randomForMod, newdata=testing)
testing_pred
```

```
##  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
```

Answers

To Submit the answers of 20 cases in the testing dataset, I have run the following code stub given on the site.

```
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(as.character(testing_pred))
```