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. 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).

DATA

The training data for this project are available here:

<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv>

The test data are available here:

<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv>

DATA LOADING

After downloading the training and testing files from the link provided, I save them to my local directory C:\PML Project. After that, I run R Studio and load the caret package along with the training and testing files saved in my local directory.

> setwd('C:/PML Project')

> library(caret)

Loading required package: lattice

Loading required package: ggplot2

Want to understand how all the pieces fit together? Buy the ggplot2 book:

http://ggplot2.org/book/

> training = read.csv("pml-training.csv",na.strings = c("NA","#DIV/0!"))

> testing = read.csv("pml-testing.csv",na.strings = c("NA","#DIV/0!"))

> dim(training)

[1] 19622   160

It can be seen that the training set has 19622 observations of 160 variables.

> table(training$classe)

   A    B    C    D    E

5580 3797 3422 3216 3607

Looking further into the training data set, it can be observed that there are 5 different classes A, B, C, D and E (classification under “classe” variable) of which class A has the highest number of observations.

After looking at the dataset, it can be seen that there are many missing values, so not all the variables will be required to predict. Further, the first 7 columns of both the training and testing datasets contain data which is not relevant for prediction, so I go ahead to remove them.

> training = training[,apply(training,2,function(x)!any(is.na(x)))]

> training = training[,-c(1:7)]

> testing = testing[,apply(testing,2,function(x)!any(is.na(x)))]

> testing = testing[,-c(1:7)]

Performing the above, we are left with 19622 observations of 53 variables.

> dim(training)

[1] 19622    53

DATA PARTITIONING

After having cleaned the training and testing data set and bringing down the variables from 160 to 53, I go ahead and divide the training set into two parts for cross validation purposes. I randomly split the data set in the ratio 75:25, the first part (75%) will be used for training purposes and the second part (25%) will be used for evaluating the model fit.

> set.seed(500000)

> inTrain = createDataPartition(y=training$classe, p=0.75, list=FALSE)

> trainset1 = training[inTrain,]

> trainset2 = training[-inTrain,]

> dim(trainset1)

[1] 14718    53

> dim(trainset2)

[1] 4904   53

Now it can be seen that the first part, dataset trainset1, consists of 14718 observations and the model will be fit on this dataset. Then the model will be evaluated on the second part, dataset trainset2, which consists of 4904 observations. Finally, the evaluated model will be applied to the testing dataset.

DATA MODELLING

For modelling, I am using Random Forest algorithm, making use of caret package. The randomForest package is loaded. Making use of the random forest model will ensure that the out of sample error is small. The 25% testing sample will be used to estimate the error. The expected error is less than 3%.

> library(randomForest)

randomForest 4.6-12

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

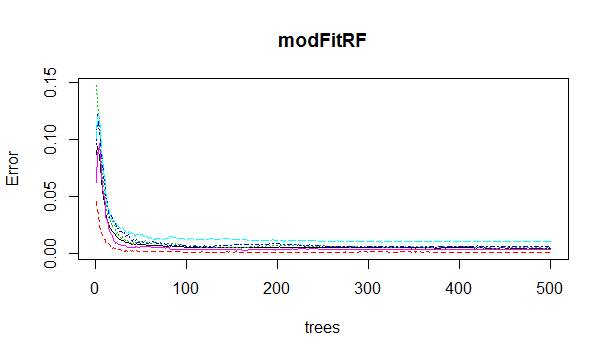
Attaching package: ‘randomForest’

The following object is masked from ‘package:ggplot2’:

    margin

> modFitRF = randomForest(classe~., data = trainset1, method="rf", importance=T, trControl = trainControl(method = "cv", classProbs = TRUE, savePredictions = TRUE, allowParallel = TRUE, number = 10))

> plot(modFitRF)



DATA MODEL RESULTS

Now after having built the model, the model is to be tested for the out of sample accuracy by applying the predict function to the “trainset2” dataset which was not used in building the model.

> predict1 = predict(modFitRF, trainset2, type = "class")

> confusionMatrix(predict1, trainset2$classe)

Confusion Matrix and Statistics

          Reference

Prediction    A    B    C    D    E

         A 1395    9    0    0    0

         B    0  939    6    0    0

         C    0    1  848    8    0

         D    0    0    1  794    4

         E    0    0    0    2  897

Overall Statistics

               Accuracy : 0.9937

                 95% CI : (0.991, 0.9957)

    No Information Rate : 0.2845

    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.992

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E

Sensitivity            1.0000   0.9895   0.9918   0.9876   0.9956

Specificity            0.9974   0.9985   0.9978   0.9988   0.9995

Pos Pred Value         0.9936   0.9937   0.9895   0.9937   0.9978

Neg Pred Value         1.0000   0.9975   0.9983   0.9976   0.9990

Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837

Detection Rate         0.2845   0.1915   0.1729   0.1619   0.1829

Detection Prevalence   0.2863   0.1927   0.1748   0.1629   0.1833

Balanced Accuracy      0.9987   0.9940   0.9948   0.9932   0.9975

After testing the random forest model, it can be seen that accuracy is 99.37%.

DATA FILE SUBMISSION

It can be seen from the confusion matrix that the model built through Random Forest method is very accurate. After testing the model with the testing dataset, it can be seen that all turned out to be rightly predicted.

> predict2 = predict(modFitRF, testing)

> predict2

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

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(predict2)