Practical Machine Learning – Weight Lifting Exercise Assignment

The objective of this exercise is to identify a suitable machine learning algorithm to predict the manner an exercise was performed (the "classe") the training set. This exercise will identify the variables to be used in the prediction model that will determine the outcome.

We will use the following steps to develop our machine learning algorithm

1. Data Exploration – Understand the data variables and features available in the dataset provided
2. Data Cleansing – Sanitize the datasets to remove unwanted or incomplete data sets
3. Model selection – Identify potential models that are suitable based on the dataset characteristics
4. Execute Models – Understand and compare results of the models to select the final model that will be used
5. Testing – Utilize the selection prediction model to cross validate the testing set and understand the accuracy

Step 1: Obtain and cleanse the data

Obtain the data and load into the memory, and then set all values with "NA","#DIV/0!","" to NA in both the training and testing dataset

|  |
| --- |
| > trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"  > testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"  > training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))  > testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!","")) |

Below is the summary of the dataset that have been loaded into the memory

|  |
| --- |
| > dim(training)  [1] 19622 160  > dim(testing)  [1] 20 160 |

Step 2 : Load the relevant libraries and perform data exploration

|  |
| --- |
| > library(rpart)  > library(rpart.plot)  > library(RColorBrewer)  > library(rattle)  Rattle: A free graphical interface for data science with R.  Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  Type 'rattle()' to shake, rattle, and roll your data.  > library(randomForest)  randomForest 4.6-14  Type rfNews() to see new features/changes/bug fixes.  Attaching package: ‘randomForest’  The following object is masked from ‘package:rattle’:  importance  The following object is masked from ‘package:ggplot2’:  margin  > set.seed(12345) |

Partitioning the training dataset into “myTraining” and “myTesting” datasets based on 60% for training and 40% for testing

|  |
| --- |
| > set.seed(12345)  > inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)  > myTraining <- training[inTrain, ]; myTesting <- training[-inTrain, ]  > dim(myTraining); dim(myTesting)  [1] 11776 160  [1] 7846 160 |

Step 3 : Perform transformation of the datasets to ensure that the model can be executed on both training and testing datasets

|  |
| --- |
| > myTraining <- myTraining[c(-1)] # remove the index/ID column |

Remove columns with too many NAs. For Variables that have more than a 60% threshold of NA’s we will remove the columns

|  |
| --- |
| > trainingV3 <- myTraining #create subset to iterate in loop  > for(i in 1:length(myTraining)) { #for every column in the training dataset  + if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .6 ) { # if n?? NAs > 60% of total observations  + for(j in 1:length(trainingV3)) {  + if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) ==1) { #if the columns are the same:  + trainingV3 <- trainingV3[ , -j] #Remove that column  + }  + }  + }  + }  > #To check the new number of observations  > dim(trainingV3)  [1] 11776 59  #Set back the myTraining dataset  > myTraining <- trainingV3  > rm(trainingV3) |

Now we will perform the transformation for myTesting and testing datasets

|  |
| --- |
| > clean1 <- colnames(myTraining)  > clean2 <- colnames(myTraining[, -59]) #already with classe column removed  > myTesting <- myTesting[clean1]  > testing <- testing[clean2]  > dim(myTesting)  [1] 7846 59  > dim(testing)  [1] 20 58 |

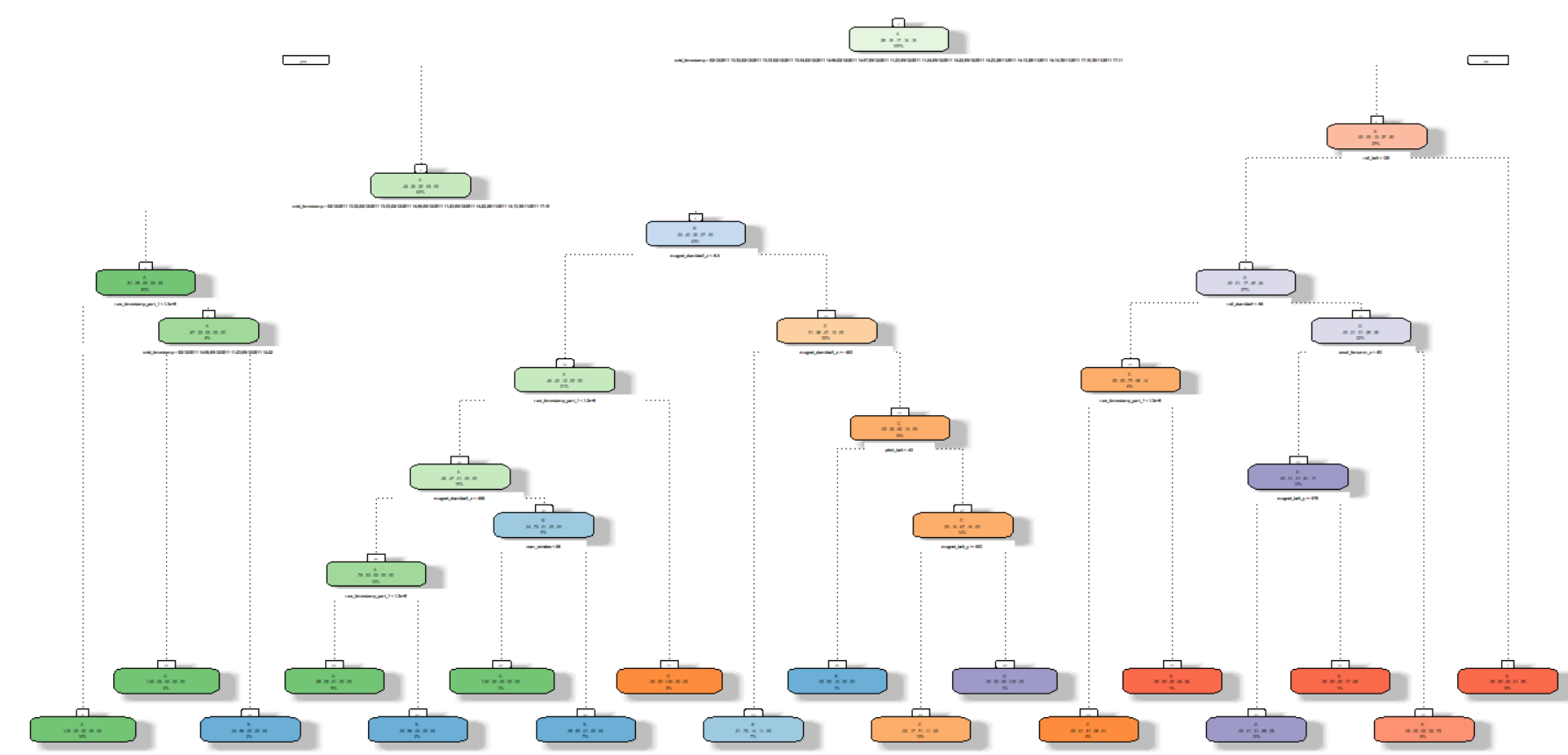
Next to coerce the myTraining and the training datasets so that the models can be executed

|  |
| --- |
| > for (i in 1:length(testing) ) {  + for(j in 1:length(myTraining)) {  + if( length( grep(names(myTraining[i]), names(testing)[j]) ) ==1) {  + class(testing[j]) <- class(myTraining[i])  + }  + }  + }  > #Ensure the Coertion worked by :  > testing <- rbind(myTraining[2, -59] , testing) #note row 2 does not mean anything, this will be removed right.. now:  > testing <- testing[-1,]  > dim(testing)  [1] 20 58  > dim(myTesting)  [1] 7846 59  > dim(myTraining)  [1] 11776 59 |

Step 3 : Perform model selection by comparing Decision tree and Random Forest

Using the caret package, leverage the rpart function that calls the decision tree algorithm

|  |
| --- |
| > modFit\_DT <- rpart(classe ~ ., data=myTraining, method="class")  > fancyRpartPlot(modFit\_DT)  > predictions\_DT <- predict(modFit\_DT, myTesting, type = "class")  > confusionMatrix(predictions\_DT, myTesting$classe)  Confusion Matrix and Statistics  Reference  Prediction A B C D E  A 2150 60 7 1 0  B 61 1260 69 64 0  C 21 188 1269 143 4  D 0 10 14 857 78  E 0 0 9 221 1360  Overall Statistics    Accuracy : 0.8789  95% CI : (0.8715, 0.8861)  No Information Rate : 0.2845  P-Value [Acc > NIR] : < 2.2e-16    Kappa : 0.8468  Mcnemar's Test P-Value : NA  Statistics by Class:  Class: A Class: B Class: C Class: D Class: E  Sensitivity 0.9633 0.8300 0.9276 0.6664 0.9431  Specificity 0.9879 0.9693 0.9450 0.9845 0.9641  Pos Pred Value 0.9693 0.8666 0.7809 0.8936 0.8553  Neg Pred Value 0.9854 0.9596 0.9841 0.9377 0.9869  Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838  Detection Rate 0.2740 0.1606 0.1617 0.1092 0.1733  Detection Prevalence 0.2827 0.1853 0.2071 0.1222 0.2027  Balanced Accuracy 0.9756 0.8997 0.9363 0.8254 0.9536  > |



Based on the result, the decision tree model from the caret package yields an accuracy of 0.8789

Now, we will use the random forest package to test the accuracy levels

|  |
| --- |
| > modFit\_RF <- randomForest(classe ~., data=myTraining)  > predictions\_RF <- predict(modFit\_RF, myTesting, type = "class")  > confusionMatrix(predictions\_RF, myTesting$classe)  Confusion Matrix and Statistics  Confusion Matrix and Statistics  Reference  Prediction A B C D E  A 2231 2 0 0 0  B 1 1516 2 0 0  C 0 0 1366 3 0  D 0 0 0 1281 0  E 0 0 0 2 1442  Overall Statistics    Accuracy : 0.9987  95% CI : (0.9977, 0.9994)  No Information Rate : 0.2845  P-Value [Acc > NIR] : < 2.2e-16    Kappa : 0.9984  Mcnemar's Test P-Value : NA  Statistics by Class:  Class: A Class: B Class: C Class: D Class: E  Sensitivity 0.9996 0.9987 0.9985 0.9961 1.0000  Specificity 0.9996 0.9995 0.9995 1.0000 0.9997  Pos Pred Value 0.9991 0.9980 0.9978 1.0000 0.9986  Neg Pred Value 0.9998 0.9997 0.9997 0.9992 1.0000  Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838  Detection Rate 0.2843 0.1932 0.1741 0.1633 0.1838  Detection Prevalence 0.2846 0.1936 0.1745 0.1633 0.1840  Balanced Accuracy 0.9996 0.9991 0.9990 0.9981 0.9998  > |

Based on the result, the random forest model yields an accuracy of 0.9987. With these accuracy results, we can then proceed to perform the testing on the “testing” dataset

Step 4: Use the selected model (RandomForest) to perform the prediction on the testing dataset

|  |
| --- |
| > predictions\_Testing\_DS <- predict(modFit\_RF, testing, type = "class")  > 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(predictions\_Testing\_DS) |

Below are the results on the testing dataset using the random forest model

1. B
2. A
3. B
4. A
5. A
6. E
7. D
8. B
9. A
10. A
11. B
12. C
13. B
14. A
15. E
16. E
17. A
18. B
19. B
20. B