PML_Final Project

Nirmal Ghimire

12/7/2020

Background

Prologue 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.

Introduction This project uses 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. They were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: "exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front" (Class E)(Velloso, Bulling, Gellersen, Ugulino, & Fuks, 2013). Read more: http://groupware.les.inf.puc-rio.br/har#ixzz6fttLBk3d

1. The goal of our project is to predict the manner in which they did the exercise.
2. The "classe" variable in the training set is the outcome variable.
3. There are variables to predict the "classe" with.

Let's kick start this project:

Invoking the required libraries

```
library(caret)
library(rpart)
library(ggplot2)
library(rpart.plot)
library(rattle)
library(randomForest)
library(corrplot)
library(partykit)
```

Getting, Preparing, and Exploring the Data

```r
trainingdata<-read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", header=TRU
testcases<-read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", header=TRUE)

#Checking the Dimension of the Data files.
dim(trainingdata)
```

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

```r
dim(testcases)
```

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

Looking at the dimensions, we see that there are total of 19,622 observations in the trainingdata and 20 in the testcases. There are total of 160 different variables, and the variable "classe" is going to be the outcome variable. Before that, I want to reduce the number of variables that have near zero values (NZV)in both trainingdata, and testcases.

```r
NZV<-nearZeroVar(trainingdata)
trainingdata<-trainingdata[, -NZV]
testcases<-testcases[, -NZV]
```

Wow!! We got rid of total of 60 variables that near zero values.

It also looks that the variables 1 through 5 are related to ID variables. So, let's get rid of them.

```r
trainingdata<-trainingdata[, -(1:5)]
testcases<-testcases[, -(1:5)]
```

As we got rid of the ID variables, we now have total of 95 variables. We can still get rid of some of the variables that have most NAs.

```r
NAs<- sapply(trainingdata, function(x) mean(is.na(x)))>0.95
trainingdata<-trainingdata[, NAs==FALSE]
testcases<-testcases[, NAs==FALSE]
```

So far, we have been able to boil down the total number of variables to 54 from 160.

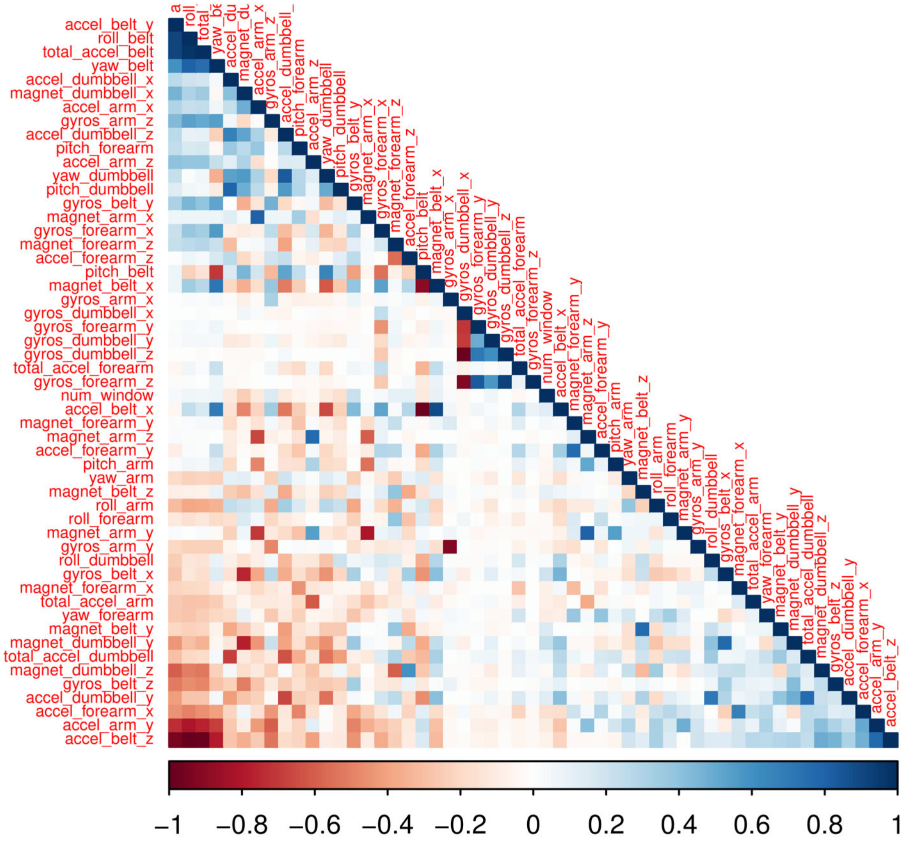
They still have a lot of, i.e., 54 variables and, so far, I don't which I want to get rid of. So, let's run a correlation and see what stands out.

```r
citation("corrplot")
```

```
##
## To cite corrplot in publications use:
##
##   Taiyun Wei and Viliam Simko (2020). R package "corrplot":
##   Visualization of a Correlation Matrix (Version 0.85). Available from
##   https://github.com/taiyun/corrplot
##
## A BibTeX entry for LaTeX users is
##
```

```
##   @Manual{corrplot2020,
##     title = {R package "corrplot": Visualization of a Correlation Matrix},
##     author = {Taiyun Wei and Viliam Simko},
##     year = {2020},
##     note = {(Version 0.85)},
##     url = {https://github.com/taiyun/corrplot},
##   }
```

```
Matrixcor<-cor(trainingdata[,-54])
corrplot(Matrixcor, type="lower", order="FPC", method="color", tl.cex=0.5)
```



Setting correlation cutoff statisticts to 0.75, I have been able to boil down the total number of variables to 21.

Training, Testing, and Validation

Now, lets break the trainingdata into training and testing sets. We are going to break total observations into 70 and 30%.

```
TrainD<-createDataPartition(trainingdata$classe, p=0.6, list=FALSE)
trainingset<-trainingdata[TrainD, ]
testset<-trainingdata[-TrainD, ]
dim(trainingset)
```

```
## [1] 11776    54
```

```r
dim(testset)
```

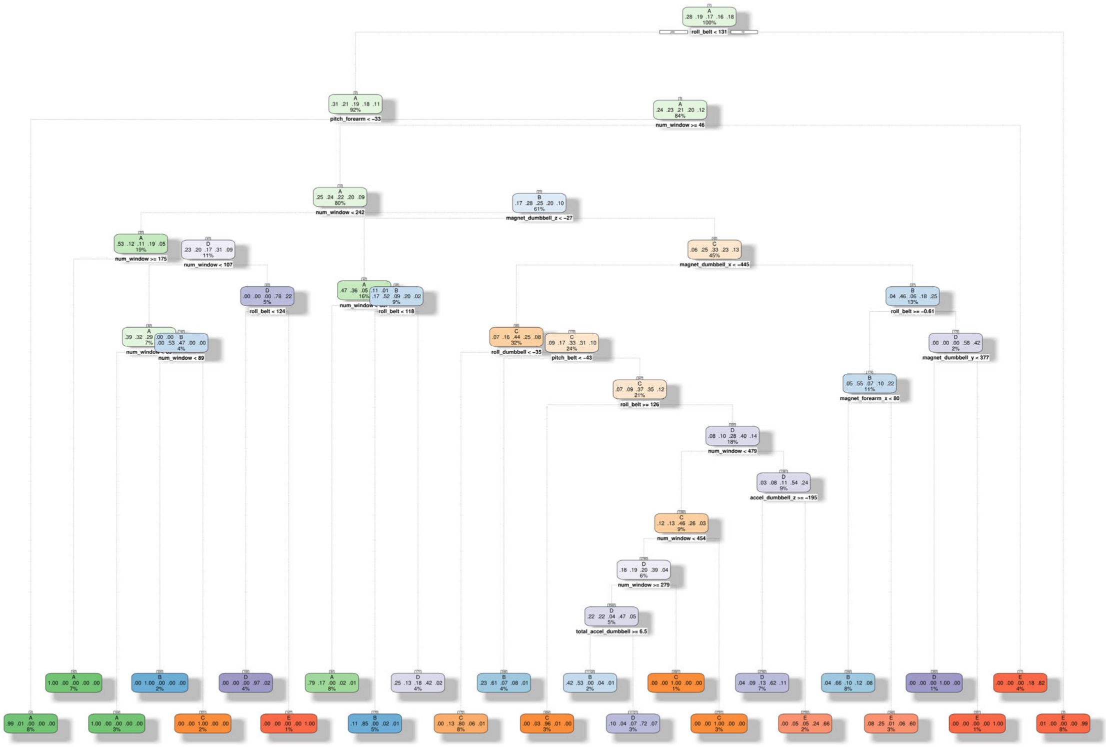
```
## [1] 7846   54
```

There are total of 11,776 observations in trainingset and 7846 observations in testset.

Fitting Models

I am going to a model on the training set.

```r
set.seed(123)
model1<-rpart(classe~., data=trainingset, method="class")
fancyRpartPlot(model1)
```



Rattle 2020–Dec–07 15:09:10 nirma

There are different number of trees and different number of interaction depths involved.

Now, let's plot the predicted result from the testing set using our model fit in the training set.

```r
predmod<-predict(model1, testset, type="class")
ctree<-confusionMatrix(predmod, as.factor(testset$classe))
ctree
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1889  109    3   20    5
##          B  194 1147   85  109   72
##          C    1   82 1142   65    6
##          D  126  101  125  963  117
##          E   22   79   13  129 1242
## 
## Overall Statistics
## 
##                Accuracy : 0.8135          
##                  95% CI : (0.8047, 0.8221)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.765           
## 
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8463   0.7556   0.8348   0.7488   0.8613
## Specificity            0.9756   0.9273   0.9762   0.9285   0.9621
## Pos Pred Value         0.9324   0.7138   0.8812   0.6725   0.8364
## Neg Pred Value         0.9411   0.9405   0.9655   0.9496   0.9686
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2408   0.1462   0.1456   0.1227   0.1583
## Detection Prevalence   0.2582   0.2048   0.1652   0.1825   0.1893
## Balanced Accuracy      0.9110   0.8415   0.9055   0.8387   0.9117
```

Looking at the results, the algorithm works good. Now, lets pass the model on the validation set.

```
valid<- predict(model1, newdata=testcases)
valid
```

```
##             A          B           C          D          E
## 1  0.00000000 1.00000000 0.000000000 0.00000000 0.00000000
## 2  0.79288026 0.17367853 0.002157497 0.02049622 0.01078749
## 3  0.42465753 0.52968037 0.000000000 0.03652968 0.00913242
## 4  1.00000000 0.00000000 0.000000000 0.00000000 0.00000000
## 5  1.00000000 0.00000000 0.000000000 0.00000000 0.00000000
## 6  0.03645200 0.09113001 0.133657351 0.62454435 0.11421628
## 7  0.03645200 0.09113001 0.133657351 0.62454435 0.11421628
## 8  0.42465753 0.52968037 0.000000000 0.03652968 0.00913242
## 9  0.98848168 0.01151832 0.000000000 0.00000000 0.00000000
## 10 0.79288026 0.17367853 0.002157497 0.02049622 0.01078749
## 11 0.03645200 0.09113001 0.133657351 0.62454435 0.11421628
## 12 0.00000000 0.00000000 1.000000000 0.00000000 0.00000000
## 13 0.08333333 0.25000000 0.005747126 0.06034483 0.60057471
## 14 0.98848168 0.01151832 0.000000000 0.00000000 0.00000000
## 15 0.00000000 0.05263158 0.052631579 0.23886640 0.65587045
## 16 0.08333333 0.25000000 0.005747126 0.06034483 0.60057471
## 17 1.00000000 0.00000000 0.000000000 0.00000000 0.00000000
## 18 0.79288026 0.17367853 0.002157497 0.02049622 0.01078749
## 19 0.00000000 1.00000000 0.000000000 0.00000000 0.00000000
## 20 0.04228330 0.66067653 0.095137421 0.11839323 0.08350951
```

The results output are helpful answering the curse project prediction quiz.