Practical Machine Learning Assignment

Surabh

7/20/2021

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, the 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

Data downloaded from: Training Data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

Testing Data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Data Load

First we download the data.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
#Set Working Directory to relevant folder
setwd("~/Coursera/Practical Machine Learning/Week 4/Assignment")
trainURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(trainURL))
testing <- read.csv(url(testURL))

dim(training)
```

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

```
dim(testing)
```

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

Exploratory Data Analysis

Then we split the training data to train and test sub-groups. We will only use the "testing" data for final validation.

```
library(caret)
indexTrain <- createDataPartition(training$classe, p = 0.7, list = FALSE)
trainSet <- training[indexTrain, ]
testSet <- training[-indexTrain, ]
```

We do all our exploratory data analysis on the train sub-group.

```
str(trainSet)
head(trainSet,n=3)
summary(trainSet)
```

We can see that there are some near zero variables and variables that contain lots of NA. We can exclude them from our prediction model.

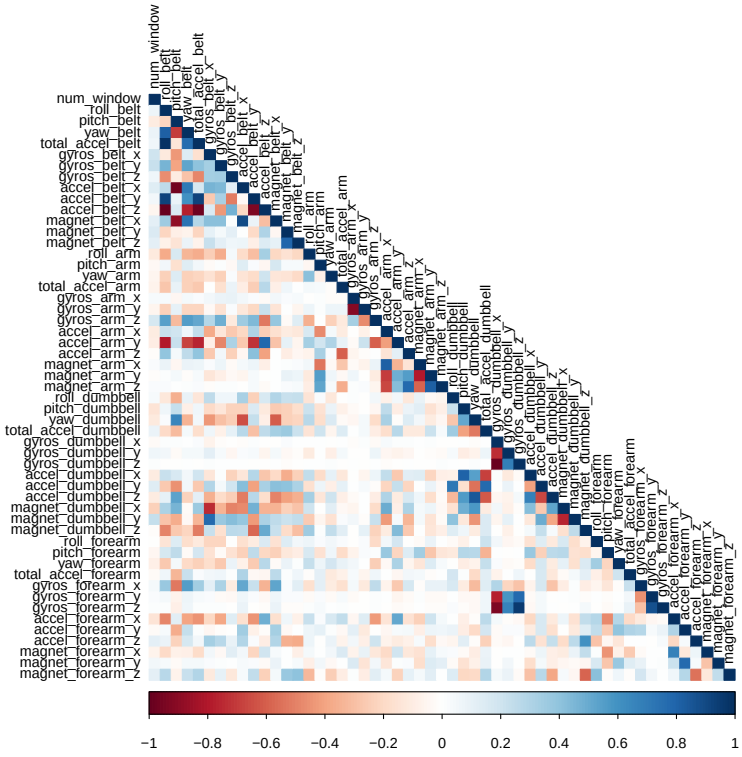
```
NZV <- nearZeroVar(trainSet)
trainSet <- trainSet[ ,-NZV]
testSet <- testSet[ ,-NZV]
label <- apply(trainSet, 2, function(x) mean(is.na(x))) > 0.95
trainSet <- trainSet[, -which(label, label == FALSE)]
testSet <- testSet[, -which(label, label == FALSE)]
trainSet <- trainSet[ , -(1:5)]
testSet <- testSet[ , -(1:5)]
```

We can now do some correlation plots.

```
library(corrplot)
```

```
## corrplot 0.90 loaded
```

```
corrMat <- cor(trainSet[,-54])
corrplot(corrMat, method = "color", type = "lower", tl.cex = 0.8, tl.col = rgb(0,0,0))
```



We could do some Principle Component Analysis to further reduce the correlated variables to improve our model in future studies.Although, there are not that many highly correlated variables so that will not impact our prediction models significantly.

Prediction Model Selection

We will predict with Decision Tree (rpart), Random Forest(rf) and Generalized Boosted Model (gbm).

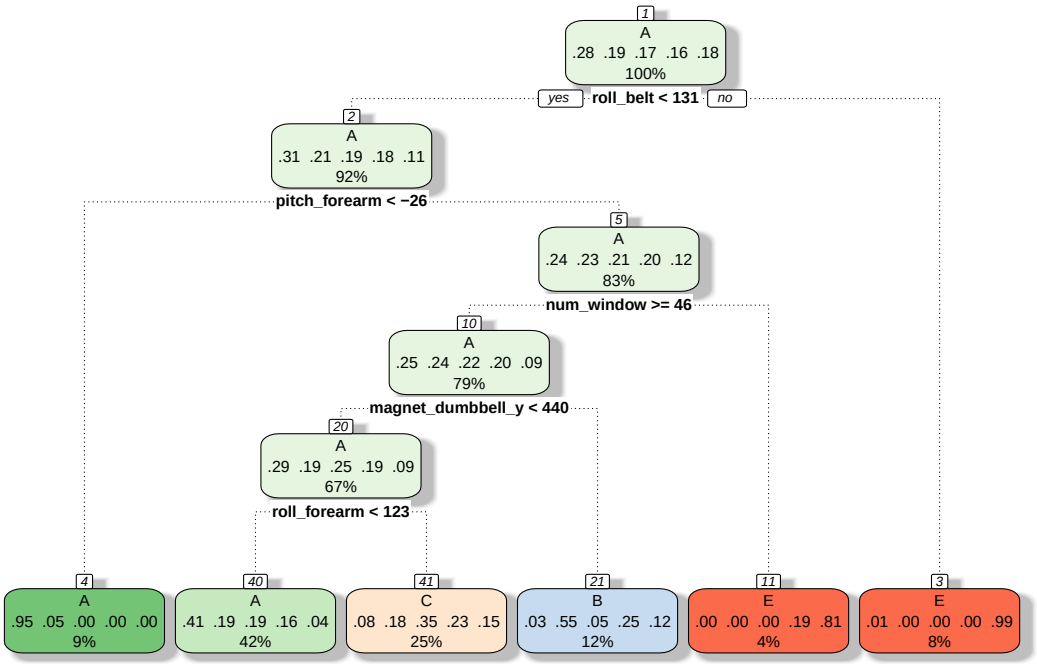
Decision Tree

```
library(caret)
library(e1071)
library(rattle)
set.seed(123)
modelDT <- train(classe ~ ., data = trainSet, method = "rpart")
modelDT$finalModel
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 12592 8694 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -26.45 1241   57 A (0.95 0.046 0 0 0) *
```

```
##      5) pitch_forearm>=-26.45 11351 8637 A (0.24 0.23 0.21 0.2 0.12)
##       10) num_window>=45.5 10850 8136 A (0.25 0.24 0.22 0.2 0.091)
##         20) magnet_dumbbell_y< 439.5 9237 6578 A (0.29 0.19 0.25 0.19 0.085)
##           40) roll_forearm< 122.5 5806 3406 A (0.41 0.19 0.19 0.16 0.044) *
##           41) roll_forearm>=122.5 3431 2221 C (0.075 0.18 0.35 0.23 0.15) *
##         21) magnet_dumbbell_y>=439.5 1613  732 B (0.034 0.55 0.046 0.25 0.12) *
##       11) num_window< 45.5 501   95 E (0 0 0 0.19 0.81) *
##    3) roll_belt>=130.5 1145    8 E (0.007 0 0 0 0.99) *
```

```
fancyRpartPlot(modelDT$finalModel)
```



Rattle 2021−Jul−20 23:45:02 surab

```
predictDT <- predict(modelDT, testSet)
confMatDT <- confusionMatrix(predictDT, testSet$classe)
confMatDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1514  484  474  410  115
##          B   25  385   34  161   79
##          C  129  270  518  342  227
##          D    0    0    0    0    0
##          E    6    0    0   51  661
##
## Overall Statistics
##
##                Accuracy : 0.523
```

```
##                  95% CI : (0.5102, 0.5359)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3775
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9044  0.33802  0.50487   0.0000   0.6109
## Specificity            0.6478  0.93700  0.80078   1.0000   0.9881
## Pos Pred Value         0.5052  0.56287  0.34859      NaN   0.9206
## Neg Pred Value         0.9446  0.85503  0.88452   0.8362   0.9185
## Prevalence             0.2845  0.19354  0.17434   0.1638   0.1839
## Detection Rate         0.2573  0.06542  0.08802   0.0000   0.1123
## Detection Prevalence   0.5093  0.11623  0.25251   0.0000   0.1220
## Balanced Accuracy      0.7761  0.63751  0.65283   0.5000   0.7995
```

Random Forest

Configure parallel processing server first. This will help run the Random Forest faster, else it runs too slow.

```
library(parallel)
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
cluster <- makeCluster(detectCores() - 1) # leave 1 core for OS
registerDoParallel(cluster)
```

```
library(caret)
set.seed(123)
controlRF <- trainControl(method = "cv", number = 3, verboseIter=FALSE, allowParallel = TRUE)
modelRF <- train(classe ~ ., data = trainSet, method = "rf", trControl = controlRF)
modelRF$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: 27
##
##         OOB estimate of  error rate: 0.25%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
```

```
## B    8 2646    4    0    0 0.0045146727
## C    0    5 2391    0    0 0.0020868114
## D    0    0   11 2240    1 0.0053285968
## E    0    0    0    4 2521 0.0015841584
```

```
predictRF <- predict(modelRF, testSet)
confMatRF <- confusionMatrix(predictRF, testSet$classe)
confMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    0    0
##          B    0 1133    2    0    0
##          C    0    1 1021    2    0
##          D    0    0    3  962    2
##          E    0    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.9975
##                  95% CI : (0.9958, 0.9986)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9951   0.9979   0.9982
## Specificity            0.9988   0.9996   0.9994   0.9990   1.0000
## Pos Pred Value         0.9970   0.9982   0.9971   0.9948   1.0000
## Neg Pred Value         1.0000   0.9987   0.9990   0.9996   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1735   0.1635   0.1835
## Detection Prevalence   0.2853   0.1929   0.1740   0.1643   0.1835
## Balanced Accuracy      0.9994   0.9972   0.9973   0.9985   0.9991
```

```
stopCluster(cluster)
registerDoSEQ()
```

Generalized Boost Model

```
library(caret)
set.seed(123)
controlGBM <- trainControl(method = "repeatedcv", number = 3, repeats = 1, verboseIter = FALSE)
modelGBM <- train(classe ~ ., data = trainSet, trControl = controlGBM, method = "gbm", verbose = FALSE)
modelGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
predictGBM <- predict(modelGBM, testSet)
confMatGBM <- confusionMatrix(predictGBM, testSet$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669   17    0    0    0
##          B    5 1108   10    7    5
##          C    0   14 1011    6    1
##          D    0    0    3  951    5
##          E    0    0    2    0 1071
##
## Overall Statistics
##
##                Accuracy : 0.9873
##                  95% CI : (0.9841, 0.99)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9839
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9728   0.9854   0.9865   0.9898
## Specificity            0.9960   0.9943   0.9957   0.9984   0.9996
## Pos Pred Value         0.9899   0.9762   0.9797   0.9917   0.9981
## Neg Pred Value         0.9988   0.9935   0.9969   0.9974   0.9977
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1883   0.1718   0.1616   0.1820
## Detection Prevalence   0.2865   0.1929   0.1754   0.1630   0.1823
## Balanced Accuracy      0.9965   0.9835   0.9905   0.9924   0.9947
```

We use a training sub group to train each of the models and test on a test sub group. We use cross validation as train control parameter in the Random Forest and GBM models. From the 3 models, we see that Random Forest has the highest accuracy on the test sub-group (~99.7%) and that is the model we will pick for our predictions as it has the best out of sample accuracy.

Final Model Fit on Validation Data

We will now fit the Random Forest model on our validation data set.

```
predictRFTest <- predict(modelRF, testing)
predictRFTest
```

```
##  [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
```