PML-Prediction

Ian Gonsalves

1/12/2021

Introduction

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, our 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 (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
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)
The test data is available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)
The data for this project comes from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Load Libraries

Note: Please **install packages** if they don't already exist on your system Loading Libraries for this project, which you should install and load them in your working environment.

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

```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

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

```
## corrplot 0.84 loaded
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:rattle':
##
##     importance
```

Reproduction of Project Outcome

We set a seed, so anyone reviewing the code can reproduce the results

```
set.seed(9999)
```

Getting Data

Setup working directory.

```
# setwd("~/Downloads/GitHub/Cour-PML/Project")
setwd("~/Downloads/PML")
```

Get dataset to the `data` folder in the current working directory.

```
trainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainFile <- "./data/pml-training.csv"
testFile  <- "./data/pml-testing.csv"
if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists(trainFile)) {
  download.file(trainUrl, destfile = trainFile, method = "curl")
}
if (!file.exists(testFile)) {
  download.file(testUrl, destfile = testFile, method = "curl")
}
rm(trainUrl)
rm(testUrl)
```

Reading Data

Load training and testing data into dataframes

```
trainData <- read.csv(trainFile)
testData <- read.csv(testFile)
dim(trainData)
```

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

```
dim(testData)
```

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

```
rm(trainFile)
rm(testFile)
```

The training data set contains 19622 observations and 160 variables, while the testing data set contains 20 observations and 160 variables. The `classe` variable in the training set is the outcome to predict.

Cleaning Data

Remove missing values / Remove insignificant variables.

1. We clean the **Near Zero Variance** Variables.

```
##                         freqRatio percentUnique zeroVar   nzv
## X                        1.000000  100.00000000   FALSE FALSE
## user_name                1.100679    0.03057792   FALSE FALSE
## raw_timestamp_part_1     1.000000    4.26562022   FALSE FALSE
## raw_timestamp_part_2     1.000000   85.53154622   FALSE FALSE
## cvtd_timestamp           1.000668    0.10192641   FALSE FALSE
## new_window              47.330049    0.01019264   FALSE  TRUE
## num_window               1.000000    4.37264295   FALSE FALSE
## roll_belt                1.101904    6.77810621   FALSE FALSE
## pitch_belt               1.036082    9.37722964   FALSE FALSE
## yaw_belt                 1.058480    9.97349913   FALSE FALSE
## total_accel_belt         1.063160    0.14779329   FALSE FALSE
## kurtosis_roll_belt    1921.600000    2.02323922   FALSE  TRUE
## kurtosis_picth_belt    600.500000    1.61553358   FALSE  TRUE
## kurtosis_yaw_belt       47.330049    0.01019264   FALSE  TRUE
## skewness_roll_belt    2135.111111    2.01304658   FALSE  TRUE
## skewness_roll_belt.1   600.500000    1.72255631   FALSE  TRUE
## skewness_yaw_belt       47.330049    0.01019264   FALSE  TRUE
## max_roll_belt            1.000000    0.99378249   FALSE FALSE
## max_picth_belt           1.538462    0.11211905   FALSE FALSE
## max_yaw_belt           640.533333    0.34654979   FALSE  TRUE
```

```
## [1] 19622   100
```

```
## [1]  20 100
```

2. Removing some columns of the dataset that do not contribute much to the accelerometer measurements.

```
regex <- grepl("^X|timestamp|user_name", names(training01))
training <- training01[, !regex]
testing <- testing01[, !regex]
rm(regex)
rm(training01)
rm(testing01)
dim(training)
```

```
## [1] 19622    95
```

```
dim(testing)
```

```
## [1] 20 95
```

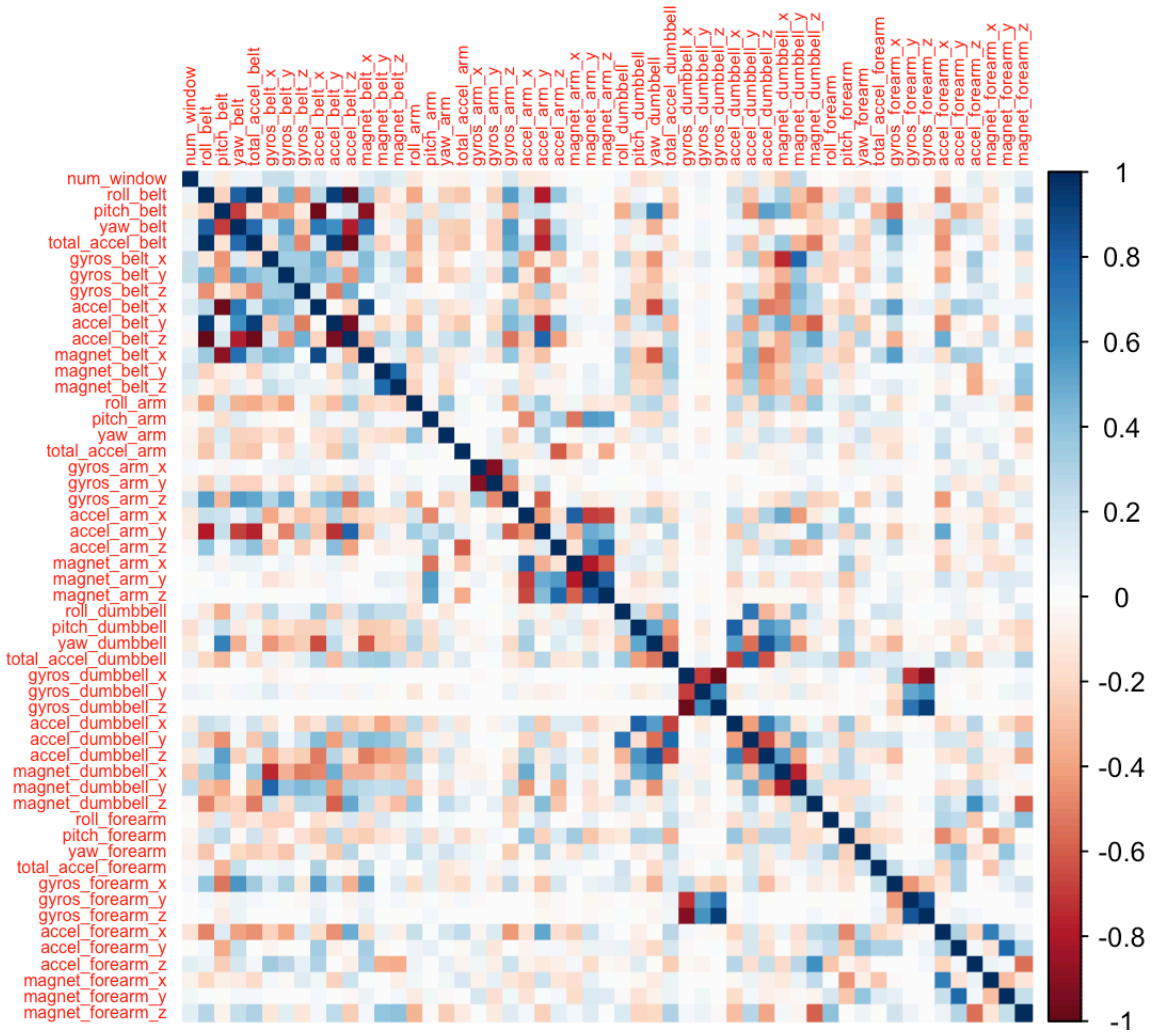
3. Removing variables with contain `NA' values.

```
cond <- (colSums(is.na(training)) == 0)
training <- training[, cond]
testing <- testing[, cond]
rm(cond)
```

Training set contains 19622 observations and 54 variables Testing set contains 20 observations and 54 variables.

Plot a matrix of the features correlation in the Training set.

```
corrplot(cor(training[, -length(names(training))]), method = "color", tl.cex = 0.5)
```



Break up our data into a Training / Validation Set

We subset into training data (75%) and a validation data (25%). Validation used for cross validation.

```
set.seed(9999)
inTrain <- createDataPartition(training$classe, p = 0.75, list = FALSE)
validation <- training[-inTrain, ]
training <- training[inTrain, ]
rm(inTrain)
```

The **Dataset** now consists of 54 variables with the observations divided as following:

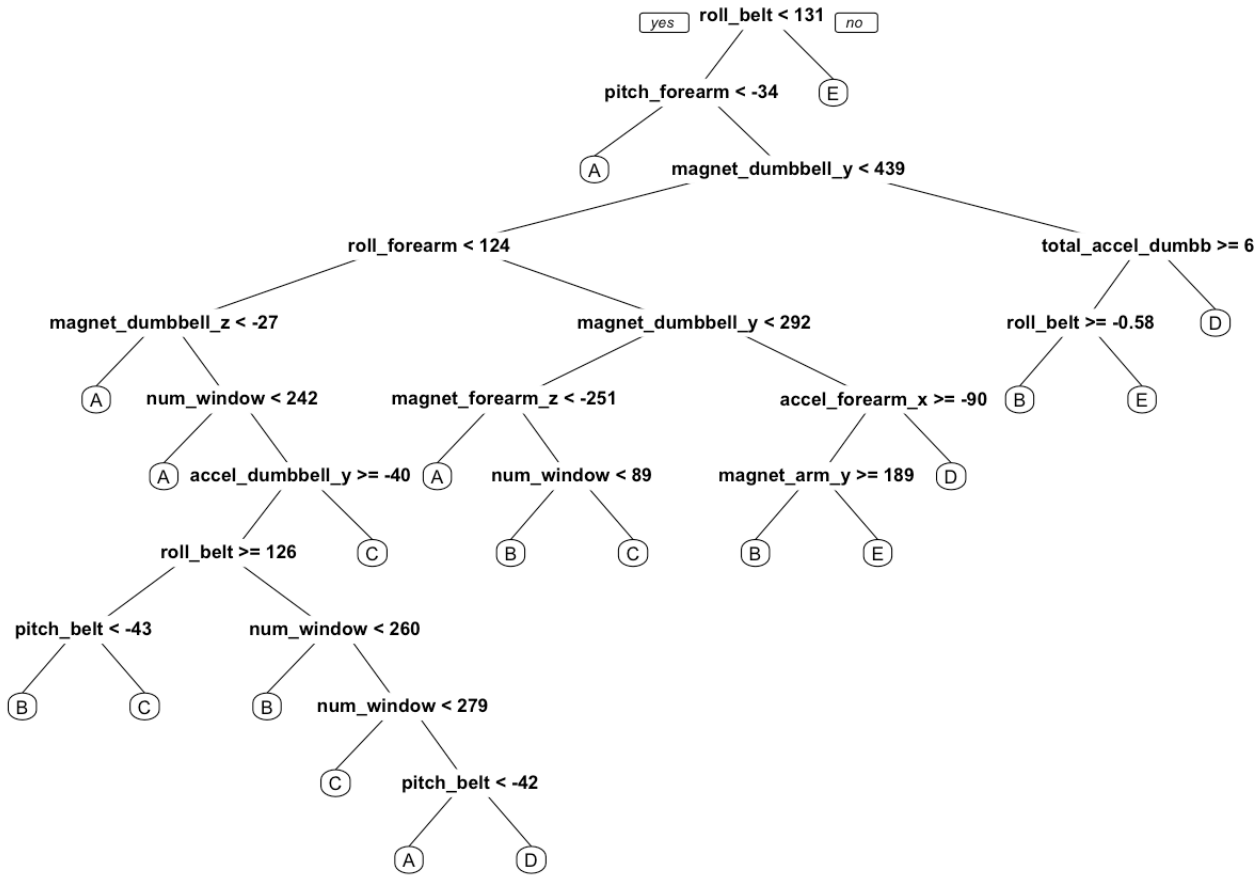
1. Training Data: 14718 observations.
2. Validation Data: 4904 observations.
3. Testing Data: 20 observations.

Predictive Modeling Phase

1: Decision Tree

Build a **Decision Tree** based Predictive Model for activity recognition.

```
modelDT <- rpart(classe ~ ., data = training, method = "class")
prp(modelDT)
```



Review performance of newly created Decision Tree prediction model on the **Validation** data.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1260   31   14   71   19
##          B  232  505   58  122   32
##          C   36   63  696   42   18
##          D   86   24  108  553   33
##          E   62   94   79   97  569
##
## Overall Statistics
##
##                Accuracy : 0.7306
##                  95% CI : (0.718, 0.743)
##     No Information Rate : 0.3418
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6572
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7518   0.7043   0.7288   0.6249   0.8480
## Specificity            0.9582   0.8940   0.9597   0.9375   0.9216
## Pos Pred Value         0.9032   0.5321   0.8140   0.6878   0.6315
## Neg Pred Value         0.8814   0.9464   0.9360   0.9190   0.9745
## Prevalence             0.3418   0.1462   0.1947   0.1805   0.1368
## Detection Rate         0.2569   0.1030   0.1419   0.1128   0.1160
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.8550   0.7991   0.8443   0.7812   0.8848
```

The Estimated Accuracy of the Random Forest Model is 73.0628059% and the Estimated Out-of-Sample Error is 26.9371941%.

2: Random Forest

We fit a predictive model for activity recognition using **Random Forest** algorithm because it automatically selects important variables and is robust to correlated covariates & outliers in general.
We will use **5-fold cross validation** when applying the algorithm.

```
modelRF <- train(classe ~ ., data = training, method = "rf", trControl = trainControl
(method = "cv", 5), ntree = 250)
modelRF
```

```
## Random Forest
##
## 14718 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11774, 11775, 11773, 11775
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9933413  0.9915765
##   27    0.9972141  0.9964762
##   53    0.9934773  0.9917491
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Now, we estimate the performance of the model on the **validation** data set.

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    1    0    0    0
##          B    1  948    0    0    0
##          C    0    1  854    0    0
##          D    0    0    1  802    1
##          E    0    0    0    3  898
##
## Overall Statistics
##
##                Accuracy : 0.9984
##                  95% CI : (0.9968, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9979   0.9988   0.9963   0.9989
## Specificity            0.9997   0.9997   0.9998   0.9995   0.9993
## Pos Pred Value         0.9993   0.9989   0.9988   0.9975   0.9967
## Neg Pred Value         0.9997   0.9995   0.9998   0.9993   0.9998
## Prevalence             0.2845   0.1937   0.1743   0.1642   0.1833
## Detection Rate         0.2843   0.1933   0.1741   0.1635   0.1831
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9995   0.9988   0.9993   0.9979   0.9991
```

```
accuracy <- postResample(predictRF, validation$classe)
sampErr <- 1 - as.numeric(confusionMatrix(validation$classe, predictRF)$overall[1])
rm(predictRF)
```

The Estimated Accuracy of the Random Forest Model is 99.8368679% and the Estimated Out-of-Sample Error is 0.1631321%.
Random Forests yielded better Results, as expected!

Perform Prediction on the Testing Data

Now, we apply the **Random Forest** model to the original testing data set downloaded from the data source. We remove the problem_id column first.

```
ncol(testing)
```

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

```
rm(accuracy)
rm(sampErr)
# predict(modelRF, testing[, -length(names(testing))])
predict(modelRF, testing[, -ncol(testing)])
```

```
##  [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
```

Output for Prediction quiz for the 20 questions

```
predictOut = function(x){
  n = length(x)
  for(i in 1:n){
    # filenames = paste0("./PredictionQuiz/Question_",i,".txt")
    filenames = paste0("./question_",i,".txt")
    write.table(x[i], file = filenames, quote = FALSE, row.names = FALSE, col.names =
FALSE)
  }
}
```

Generating the Files.

```
predictOut(predict(modelRF, testing[, -ncol(testing)]))
rm(modelRF)
rm(training)
rm(testing)
rm(validation)
rm(predictOut)
```