Practical Machine Learning Project

Soumadip Roy

February 11, 2018

Prediction Assignment

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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data Processing

Load appropriate packages for the work to be performed.

```
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(corrplot)
library(randomForest)
```

Making Data Ready

Downloading the two files and reading the files and performing the initial checks on the files so that they can be processed after being read as CSV files into R

```
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="libcurl")
}
if (!file.exists(testFile)) {
  download.file(testUrl, destfile=testFile, method="libcurl")
}

train <- read.csv("./data/pml-training.csv")
test <- read.csv("./data/pml-testing.csv")
dim(train)
```

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

```
dim(test)
```

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

The training data set contains 19622 observations and 160 variables for each, while the testing data set contains 20 observations and 160 variables for each. The “classe” variable in the training set is the outcome to predict and thus build the model.

Cleaning Data

Determining the levels for “classe” field giving insights into the data.

```
class.level = train[, "classe"]
class.levels = class.level
levels(class.levels)
```

```
## [1] "A" "B" "C" "D" "E"
```

Next steps:

Taking out classe field so thast it can be added later after cleaning up of data is done

```
train.classe <- train$classe
```

As has been mentioned in the assignment that one only has to use data from accelerometers on the belt, forearm, arm, and dumbell, so the features are extracted based on these keywords. Thus 160 variables are not all required for the analysis and taking an important step towards building a better model.

```
filter = grepl("belt|arm|dumbell", names(train))
trainfltrd = train[, filter]
testfltrd = test[, filter]

dim(trainfltrd)
```

```
## [1] 19622   114
```

```
dim(testfltrd)
```

```
## [1]  20 114
```

All columns having NA missing values are removed so that we have a (taking test data in the previous step as reference)

```
cols.without.na = colSums(is.na(testfltrd)) == 0
trainclean <- trainfltrd[, cols.without.na]
testclean <- testfltrd[, cols.without.na]
```

Converting all the fields/features/predictors to numeric so that Random Forest function can be applied later in the Data MOdelling phase

```
trainclean.num <- as.data.frame(lapply(trainclean, as.numeric))
testclean.num <- as.data.frame(lapply(testclean, as.numeric))
```

Finalizing the train dataset by adding the classe field to it for model fitment

```
trainclean.num$classe <- train.classe

dim(trainclean.num)
```

```
## [1] 19622    40
```

```
dim(testclean.num)
```

```
## [1] 20 39
```

Pre Processing

So we end up with 39 relevant columns(features) in the data which we would pass through preprocessing and build the data modelling on. There are 5 classes in the outcome “A” “B” “C” “D” “E”

Check with nearZeroVar function

```
zero.var <- nearZeroVar(trainclean.num, saveMetrics = TRUE)
zero.var
```

```
##                     freqRatio percentUnique zeroVar   nzv
## roll_belt            1.101904     6.7781062   FALSE FALSE
## pitch_belt           1.036082     9.3772296   FALSE FALSE
## yaw_belt             1.058480     9.9734991   FALSE FALSE
## total_accel_belt     1.063160     0.1477933   FALSE FALSE
## gyros_belt_x         1.058651     0.7134849   FALSE FALSE
## gyros_belt_y         1.144000     0.3516461   FALSE FALSE
## gyros_belt_z         1.066214     0.8612782   FALSE FALSE
## accel_belt_x         1.055412     0.8357966   FALSE FALSE
## accel_belt_y         1.113725     0.7287738   FALSE FALSE
## accel_belt_z         1.078767     1.5237998   FALSE FALSE
## magnet_belt_x        1.090141     1.6664968   FALSE FALSE
## magnet_belt_y        1.099688     1.5187035   FALSE FALSE
## magnet_belt_z        1.006369     2.3290184   FALSE FALSE
## roll_arm            52.338462    13.5256345   FALSE FALSE
## pitch_arm           87.256410    15.7323412   FALSE FALSE
## yaw_arm             33.029126    14.6570176   FALSE FALSE
## total_accel_arm      1.024526     0.3363572   FALSE FALSE
## gyros_arm_x          1.015504     3.2769341   FALSE FALSE
## gyros_arm_y          1.454369     1.9162165   FALSE FALSE
## gyros_arm_z          1.110687     1.2638875   FALSE FALSE
## accel_arm_x          1.017341     3.9598410   FALSE FALSE
## accel_arm_y          1.140187     2.7367241   FALSE FALSE
## accel_arm_z          1.128000     4.0362858   FALSE FALSE
## magnet_arm_x         1.000000     6.8239731   FALSE FALSE
## magnet_arm_y         1.056818     4.4439914   FALSE FALSE
## magnet_arm_z         1.036364     6.4468454   FALSE FALSE
## roll_forearm        11.589286    11.0895933   FALSE FALSE
## pitch_forearm       65.983051    14.8557741   FALSE FALSE
## yaw_forearm         15.322835    10.1467740   FALSE FALSE
## total_accel_forearm  1.128928     0.3567424   FALSE FALSE
## gyros_forearm_x      1.059273     1.5187035   FALSE FALSE
## gyros_forearm_y      1.036554     3.7763735   FALSE FALSE
## gyros_forearm_z      1.122917     1.5645704   FALSE FALSE
## accel_forearm_x      1.126437     4.0464784   FALSE FALSE
## accel_forearm_y      1.059406     5.1116094   FALSE FALSE
## accel_forearm_z      1.006250     2.9558659   FALSE FALSE
## magnet_forearm_x     1.012346     7.7667924   FALSE FALSE
## magnet_forearm_y     1.246914     9.5403119   FALSE FALSE
## magnet_forearm_z     1.000000     8.5771073   FALSE FALSE
## classe               1.469581     0.0254816   FALSE FALSE
```

There are no predictors without variability. So there is feature or predictor to be removed further.

Building a Model

We would try to fit Random Forest algorithm because it automatically selects important variables and is robust to correlated covariates & outliers in general.

```
set.seed(22111)
validsample <- createDataPartition(trainclean.num$classe, p=0.7, list = F)
sampleData <- trainclean.num[-validsample,]
dim(sampleData)
```

```
## [1] 5885   40
```

Running to generate model on 70% of train data , rest 30% is left for Validation

```
trainclean.num <- trainclean.num[validsample,]
dim(trainclean.num)
```

```
## [1] 13737    40
```

```
output.forest <- randomForest(classe ~ ., data = trainclean.num)
output.forest
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = trainclean.num)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 0.85%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3900    2    2    2    0 0.001536098
## B   16 2632    9    0    1 0.009781791
## C    2   31 2350   13    0 0.019198664
## D    0    1   31 2216    4 0.015985790
## E    0    0    0    3 2522 0.001188119
```

Measuring accuracy using the Validation dataset

```
predictrf <- predict(output.forest,sampleData )
confusionMatrix(sampleData$classe,predictrf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    6 1126    7    0    0
##          C    0    8 1014    4    0
##          D    0    0   13  951    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9911, 0.9954)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9918
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9929   0.9807   0.9958   1.0000
## Specificity            1.0000   0.9973   0.9975   0.9974   1.0000
## Pos Pred Value         1.0000   0.9886   0.9883   0.9865   1.0000
## Neg Pred Value         0.9986   0.9983   0.9959   0.9992   1.0000
## Prevalence             0.2855   0.1927   0.1757   0.1623   0.1839
## Detection Rate         0.2845   0.1913   0.1723   0.1616   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9982   0.9951   0.9891   0.9966   1.0000
```

```
accuracy <- postResample(predictrf,sampleData$classe)
oose <- 1- as.numeric(confusionMatrix(sampleData$classe,predictrf)$overall[1])
```

Accurancy and OOSE

```
accuracy
```

```
##  Accuracy     Kappa
## 0.9935429 0.9918315
```

```
oose
```

```
## [1] 0.006457094
```

Predicting for Test data set

```
prdoutput <- predict(output.forest,testclean.num)
prdoutput
```

```
##  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
```