Machine Learning Prediction Assignment

Shubhendu Dash

17/07/2020

1. Introduction

Utilizing gadgets, for example, Jawbone Up, Nike FuelBand, and Fitbit it is currently conceivable to gather a lot of information about close to home action moderately modestly. These sort of gadgets are a piece of the evaluated self development – a gathering of aficionados who take estimations about themselves routinely to improve their wellbeing, to discover designs in their conduct, or on the grounds that they are tech nerds. One thing that individuals routinely do is measure the amount of a specific movement they do, however they once in a while evaluate how well they do it.

In this undertaking, we will utilize information from accelerometers on the belt, lower arm, arm, and dumbell of 6 members to anticipate the way in which they did the activity.

2. Data Preprocessing

2.1 Loading required packages from library

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2
```

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

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

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.0.2
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.0.2
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.2
```

```
## 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
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.0.2

## corrplot 0.84 loaded
```

```
library(doParallel)
```

```
## Warning: package 'doParallel' was built under R version 4.0.2

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 4.0.2

## Loading required package: iterators

## Warning: package 'iterators' was built under R version 4.0.2

## Loading required package: parallel
```

2.2 Downloading the Data

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

2.3 Reading the Data

After downloading the data from the data source, we can read the two csv files into two different data frames.

```
trainRaw <- read.csv("./data/pml-training.csv", na.strings = c("NA", "", "#DIV/0!"))
testRaw <- read.csv("./data/pml-testing.csv", na.strings = c("NA", "", "#DIV/0!"))
dim(trainRaw)
```

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

```
dim(testRaw)
```

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

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.

2.4 Cleaning the data

In this step, we will clean the data and get rid of observations with missing values as well as some meaningless variables.

```
sum(complete.cases(trainRaw))
```

```
## [1] 0
```

First, we will remove the columns that contain NA missing values.

```
trainRaw <- trainRaw[, colSums(is.na(trainRaw)) == 0]
testRaw <- testRaw[, colSums(is.na(testRaw)) == 0]
dim(trainRaw)
```

```
## [1] 19622    60
```

```
dim(testRaw)
```

```
## [1] 20 60
```

Next, we will get rid of some columns that do not contribute much to the accelerometer measurements.

```
classe <- trainRaw$classe
trainRemove <- grepl("^X|timestamp|window", names(trainRaw))
trainRaw <- trainRaw[, !trainRemove]
trainCleaned <- trainRaw[, sapply(trainRaw, is.numeric)]
trainCleaned$classe <- classe
testRemove <- grepl("^X|timestamp|window", names(testRaw))
testRaw <- testRaw[, !testRemove]
testCleaned <- testRaw[, sapply(testRaw, is.numeric)]
dim(trainCleaned)
```

```
## [1] 19622    53
```

Now, the cleaned training data set contains 19622 observations and 53 variables, while the testing data set contains 20 observations and 53 variables. The "classe" variable is still in the cleaned training set.

Now checking for near zero values in training dataset.

```
trainNZV <- nzv(trainCleaned[, -ncol(trainCleaned)], saveMetrics = TRUE)
rownames(trainNZV)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"
```

```
dim(trainNZV)
```

```
## [1] 52  4
```

2.5 Slicing the data

Then, we can split the cleaned training set into a pure training data set (60%) and a validation data set (40%). We will use the validation data set to conduct cross validation in future steps.

```
set.seed(22519) # For reproducibile purpose
inTrain <- createDataPartition(trainCleaned$classe, p = 0.6, list = FALSE)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
dim(trainData)
```

```
## [1] 11776    53
```

```
dim(testData)
```

```
## [1] 7846   53
```

3. Data Modeling

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.

```
myModelFilename <- "myModel.RData"
if (!file.exists(myModelFilename)) {
    #Parallel cores
    #Require(parallel)
    library(doParallel)
    ncores <- makeCluster(detectCores() - 1)
    registerDoParallel(cores=ncores)
    getDoParWorkers() # 3

    # use Random Forest method with Cross Validation, 4 folds
    myModel <- train(classe ~ .
                , data = trainData
                , method = "rf"
                , metric = "Accuracy"
    # categorical outcome variable so choose accuracy
                , preProcess=c("center", "scale")
    # attempt to improve accuracy by normalising
                , trControl=trainControl(method = "cv"
                                         , number = 4
                                         # folds of the training data
                                         , p= 0.60
                                         , allowParallel = TRUE
#                                        , seeds=NA
                                         # don't let workers set seed
                                         )
                )
    save(myModel, file = "myModel.RData")
    # 3:42 .. 3:49 without preProcess
    # 3:51 .. 3:58 with preProcess
    stopCluster(ncores)
} else {
    # Use cached model
    load(file = myModelFilename, verbose = TRUE)
}
```

```
## Loading objects:
##   myModel
```

```
print(myModel, digits = 4)
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (52), scaled (52)
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 8834, 8831, 8831, 8832
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9878    0.9845
```

```
##   27    0.9880    0.9849
##   52    0.9766    0.9704
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

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

```
#Predicting
predictRf <- predict(myModel, newdata = testData)
#Testing accuracy
confusionMatrix(table(predictRf, testData$classe))
```

```
## Confusion Matrix and Statistics
##
##
## predictRf    A    B    C    D    E
##         A 2229   10    0    0    0
##         B    0 1502    7    0    0
##         C    3    6 1359   20    5
##         D    0    0    2 1260    2
##         E    0    0    0    6 1435
##
## Overall Statistics
##
##                Accuracy : 0.9922
##                  95% CI : (0.99, 0.994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9902
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9895   0.9934   0.9798   0.9951
## Specificity            0.9982   0.9989   0.9948   0.9994   0.9991
## Pos Pred Value         0.9955   0.9954   0.9756   0.9968   0.9958
## Neg Pred Value         0.9995   0.9975   0.9986   0.9960   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1914   0.1732   0.1606   0.1829
## Detection Prevalence   0.2854   0.1923   0.1775   0.1611   0.1837
## Balanced Accuracy      0.9984   0.9942   0.9941   0.9896   0.9971
```

```
accuracy <- postResample(table(predictRf), table(testData$classe))
accuracy
```

```
##       RMSE   Rsquared        MAE
## 15.7480157  0.9980289 12.8000000
```

```
oose <- 1 - as.numeric(confusionMatrix(table(testData$classe, predictRf))$overall[1])
oose
```

```
## [1] 0.007774662
```

So, the estimated accuracy of the model is 99.80% and the estimated out-of-sample error is 0.77%.

5. Final model data and important predictors

```
myModel$finalModel
```

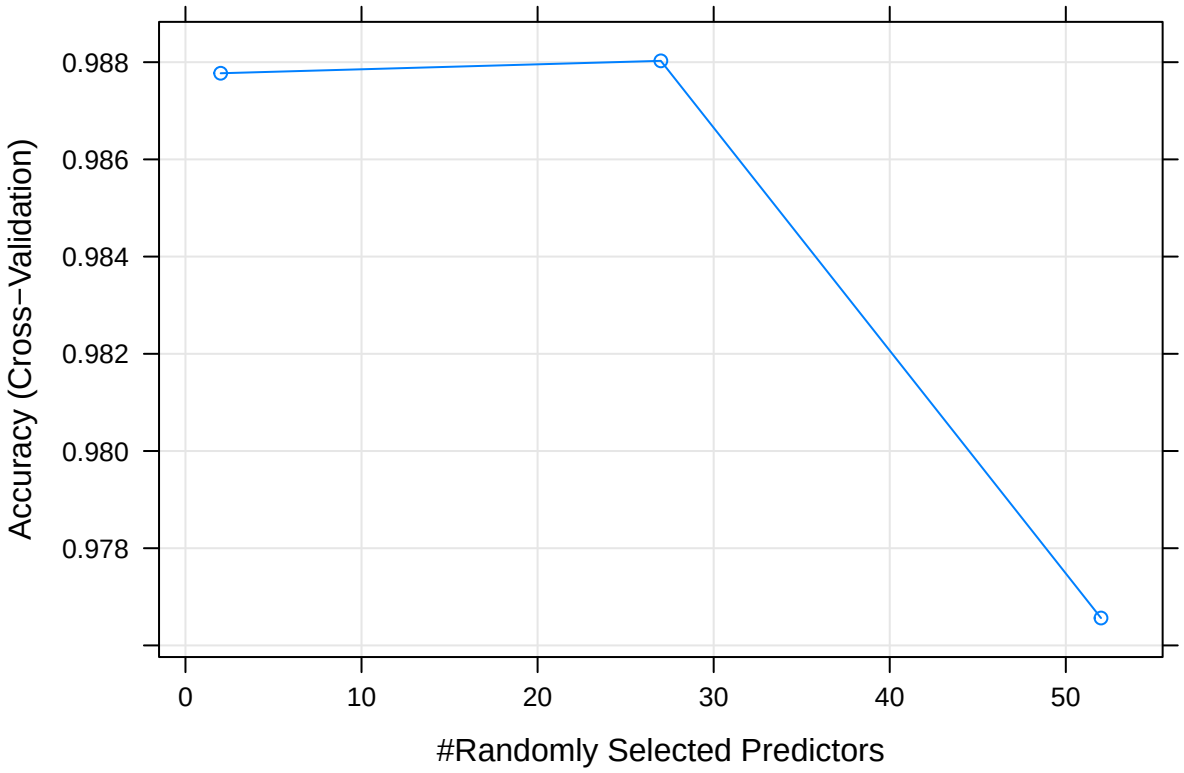
```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.86%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3340    5    2    0    1 0.002389486
## B   18 2250   11    0    0 0.012724879
## C    0   15 2029   10    0 0.012171373
## D    0    0   22 1903    5 0.013989637
## E    0    1    3    8 2153 0.005542725
```

```
varImp(myModel)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt            100.000
## pitch_forearm         61.164
## yaw_belt              52.086
## magnet_dumbbell_y     44.697
## pitch_belt            44.004
## roll_forearm          42.328
## magnet_dumbbell_z     41.289
## accel_dumbbell_y      19.333
## roll_dumbbell         17.774
## accel_forearm_x       17.159
## magnet_dumbbell_x     16.059
## magnet_belt_z         15.639
## magnet_forearm_z      14.573
## accel_dumbbell_z      14.034
## total_accel_dumbbell  14.004
## magnet_belt_y         13.357
```

```
## accel_belt_z          11.769
## yaw_arm               11.572
## gyros_belt_z          11.289
## magnet_belt_x          9.287
```

```
plot(myModel)
```



27 variables were tried at each split and the reported OOB Estimated Error is a low 0.86%.

Overall we have sufficient confidence in the prediction model to predict classe for the 20 quiz/test cases.

4. Predicting for Test Data Set

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

```
result <- predict(myModel, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [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
```

5. Appendix: Figures

Figure 1: Correlation Matrix Visualization

```
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, method="color")
```

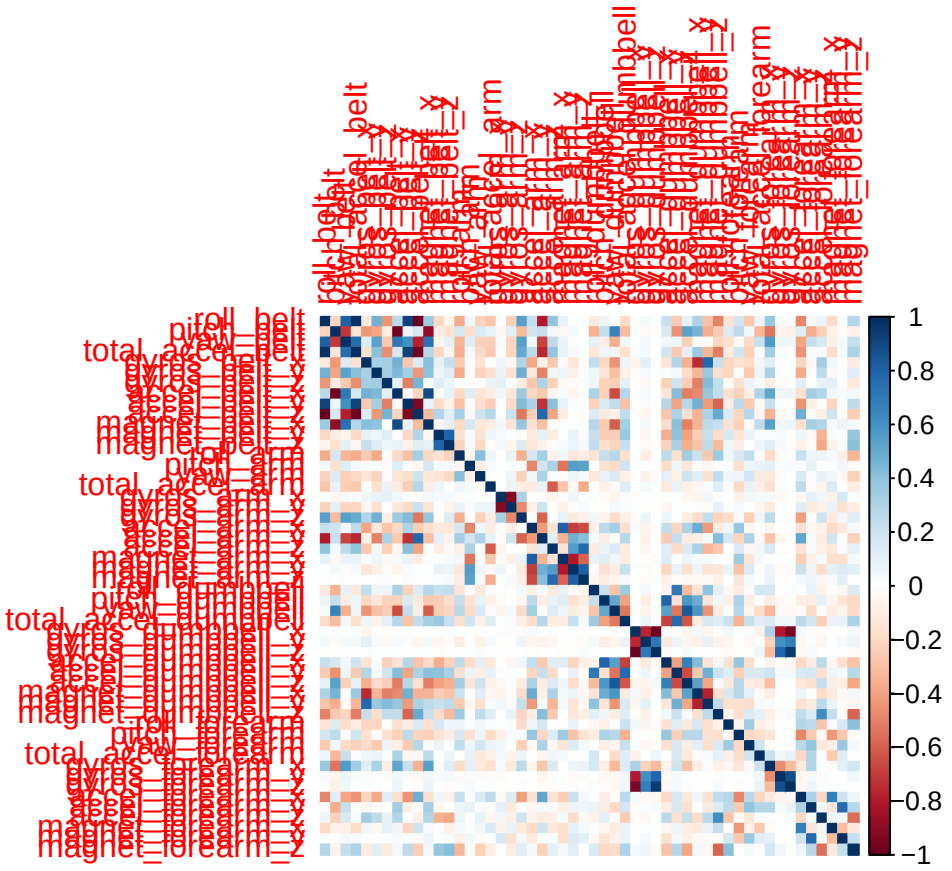
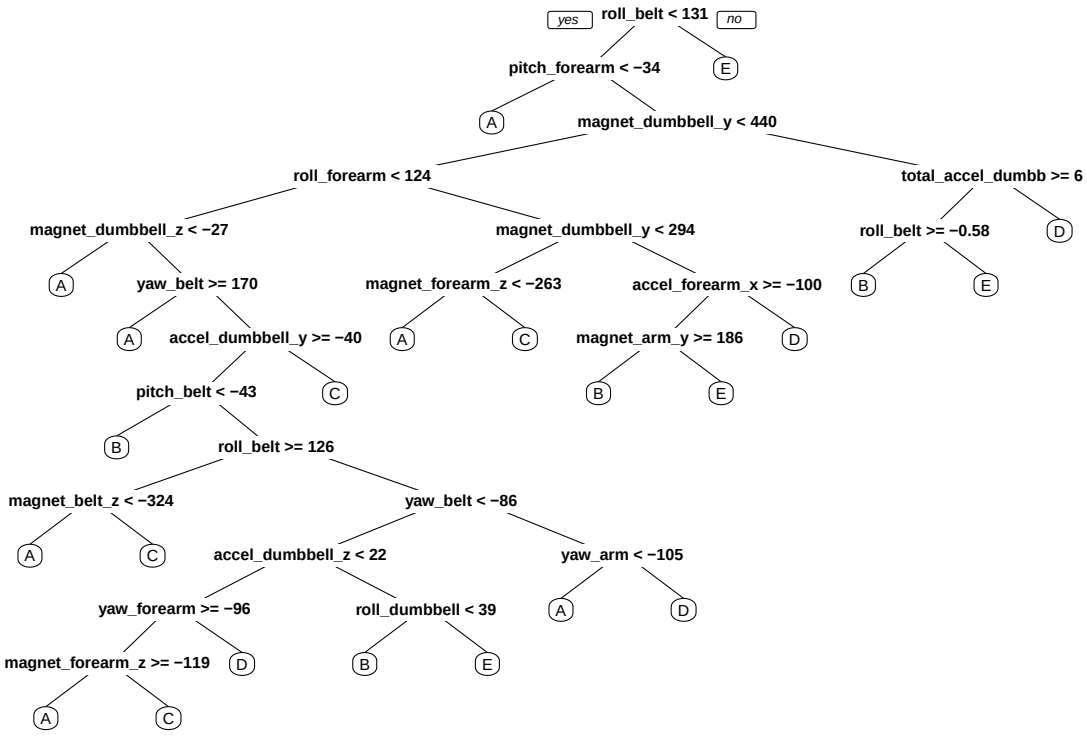


Figure 2: Decision Tree Visualization

```
treeModel <- rpart(classe ~ ., data=trainData, method="class")
prp(treeModel) # fast plot
```


yes
roll_belt < 131
no
pitch_forearm < −34
E
A
magnet_dumbbell_y < 440
roll_forearm < 124
total_accel_dumbb >= 6
magnet_dumbbell_z < −27
magnet_dumbbell_y < 294
roll_belt >= −0.58
D
A
yaw_belt >= 170
magnet_forearm_z < −263
accel_forearm_x >= −100
B
E
A
accel_dumbbell_y >= −40
A
C
magnet_arm_y >= 186
D
pitch_belt < −43
C
B
E
B
roll_belt >= 126
magnet_belt_z < −324
yaw_belt < −86
A
C
accel_dumbbell_z < 22
yaw_arm < −105
yaw_forearm >= −96
roll_dumbbell < 39
A
D
magnet_forearm_z >= −119
D
B
E
A
C