Project

Arturo Rosas Azañero

13/9/2020

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.

The data is taken 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. The goal of this project is to predict the manner in which the participants did the exercise.

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

Data Processing

Pre-processing/set-up

Necessary packages are loaded

```
library(caret)
library(rattle)
```

Data download and read

```
## Download and unzip the data, provided it doesn't already exist
if(!file.exists("pml-training.csv")){
  fileUrl<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
  download.file(fileUrl,destfile="pml-training.csv",method="curl")
}
if(!file.exists("pml-testing.csv")){
  fileUrl<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
  download.file(fileUrl,destfile="pml-testing.csv",method="curl")
}
## Read in the data
if(!"trainIn" %in% ls()){
trainIn <- read.csv("pml-training.csv")
}
if(!"testIn" %in% ls()){
testIn <- read.csv("pml-testing.csv")
}
```

```
dim(trainIn)
```

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

```
dim(testIn)
```

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

```
str(trainIn)
```

As can be seen, there are 19622 observations from 160 variables in the training data set; while there are 20 observations in the testing set.

Cleaning data

Both the training and test data sets need to be trimmed down - the first seven variables do not affect the 'classe' variable and therefore are removed; along with variables with a majority of NA values, and variables that are near-zero-variance.

```
train <- trainIn[,-c(1:7)]
trainNZV <- nearZeroVar(train)
train <- train[,-trainNZV]
train <- train[, colSums(is.na(train)) == 0]
test <- testIn[,-c(1:7)]
testNZV <- nearZeroVar(test)
test <- test[,-testNZV]
test <- test[, colSums(is.na(test)) == 0]
dim(train)
```

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

```
dim(test)
```

```
## [1] 20 53
```

```
## Check to see if data contains missing values
anyNA(train)
```

```
## [1] FALSE
```

The data sets have now been reduced to 53 variables.

Model

Two models will be analysed: Decision Tree and Random Forest.

The data is partitioned to create a 60% training set and a 40% test set.

```
## Data slicing
set.seed(5)
inTrain <- createDataPartition(train$classe, p=0.6, list=FALSE)
training <- train[inTrain,]
testing <- train[-inTrain,]
```

Cross validation is used as the resampling method and this is set using the 'trainControl' function. The number of resampling iterations will be set at 3.

```
trctrl <- trainControl(method="cv", number=3)
```

Prediction with classification trees

```
## Training Decision Tree classifier denoted by 'rpart'
modelTree <- train(classe ~ ., method="rpart", trControl=trctrl, data=training)
```

The dendogram can be seen in Appendix 1.

The model is then validated by running the *testing* data on it.

```
# display confusion matrix and model accuracy
trainPrTr <- predict(modelTree, testing)
cmTree <- confusionMatrix(table(testing$classe, trainPrTr))
cmTree
```

```
## Confusion Matrix and Statistics
##
##    trainPrTr
##        A    B    C    D    E
##   A 2024   41  159    0    8
##   B  632  504  382    0    0
##   C  640   39  689    0    0
##   D  571  241  474    0    0
##   E  205  209  379    0  649
##
## Overall Statistics
##
##                Accuracy : 0.4927
##                  95% CI : (0.4816, 0.5039)
##     No Information Rate : 0.519
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3371
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4971  0.48743  0.33077       NA  0.98782
## Specificity            0.9449  0.85115  0.88218   0.8361  0.88969
## Pos Pred Value         0.9068  0.33202  0.50365       NA  0.45007
## Neg Pred Value         0.6352  0.91625  0.78481       NA  0.99875
## Prevalence             0.5190  0.13179  0.26549   0.0000  0.08374
## Detection Rate         0.2580  0.06424  0.08782   0.0000  0.08272
## Detection Prevalence   0.2845  0.19347  0.17436   0.1639  0.18379
## Balanced Accuracy      0.7210  0.66929  0.60648       NA  0.93876
```

```
# Calculation of accuracy and out of sample error
accTree <- sum(trainPrTr == testing$classe)/length(trainPrTr)
ooseTree <- 1 - accTree
```

Prediction with Random Forest

```
## Training Random Forest denoted by 'rf'
modelRF <- train(classe ~ ., method="rf", trControl=trctrl, data = training)
```

The model is then validated by running the *testing* data on it.

```
# display confusion matrix and model accuracy
trainPrRF <- predict(modelRF, testing)
cmRF <- confusionMatrix(table(testing$classe, trainPrRF))
cmRF
```

```
## Confusion Matrix and Statistics
##
##    trainPrRF
##        A    B    C    D    E
##   A 2226    6    0    0    0
##   B    9 1499   10    0    0
##   C    0    3 1360    5    0
##   D    0    1   25 1256    4
##   E    0    1    2    6 1433
##
## Overall Statistics
##
##                Accuracy : 0.9908
##                  95% CI : (0.9885, 0.9928)
##     No Information Rate : 0.2849
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9884
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9960   0.9927   0.9735   0.9913   0.9972
## Specificity            0.9989   0.9970   0.9988   0.9954   0.9986
## Pos Pred Value         0.9973   0.9875   0.9942   0.9767   0.9938
## Neg Pred Value         0.9984   0.9983   0.9943   0.9983   0.9994
## Prevalence             0.2849   0.1925   0.1781   0.1615   0.1832
## Detection Rate         0.2837   0.1911   0.1733   0.1601   0.1826
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9975   0.9949   0.9861   0.9934   0.9979
```

```
# Calculation of accuracy and out of sample error
accRF <- sum(trainPrRF == testing$classe)/length(trainPrRF)
ooseRF <- 1 - accRF
```

A plot showing the accuracy of the model by the number of predictors used can be seen in Appendix 2. This shows that the number of predictors that gives the highest accuracy is 27. In addition, using the varImp function (Appendix 3), the 20 most important variables are shown. The variable *roll_belt* has the highest importance, meaning that its impact on the outcome values is significant.

Results

```
print(data.frame(
    "Model" = c('Classification Tree', 'Random Forest'),
    "Accuracy" = c(accTree, accRF),
    "Out of Sample Error" = c(ooseTree, ooseRF)), digits = 3)
```

```
##                 Model Accuracy Out.of.Sample.Error
## 1 Classification Tree    0.493             0.50726
## 2       Random Forest    0.991             0.00918
```

The table above shows that the **Random Forest** model has the highest accuracy of the two models, and by extension, the lowest out of sample error, therefore it will be used on the test data.

```
# Prediction of new values
final <- predict(modelRF, newdata=test)
final
```

```
##  [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
```

Appendix

Appendix 1

```
fancyRpartPlot(modelTree$finalModel)
```

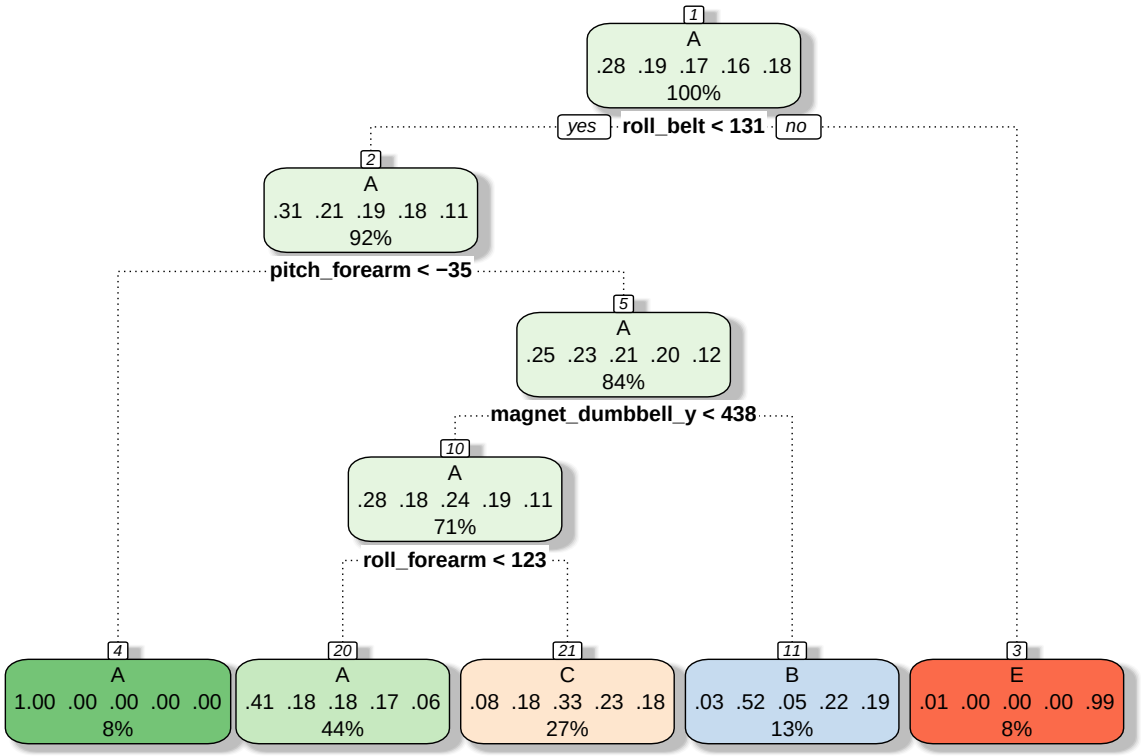

Figure 1: Figure 1: Decision Tree

Appendix 2

```
plot(modelRF)
```

Appendix 3

```
varImp(modelRF)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt            100.000
## pitch_forearm         59.313
## yaw_belt              49.608
## magnet_dumbbell_y     43.629
```

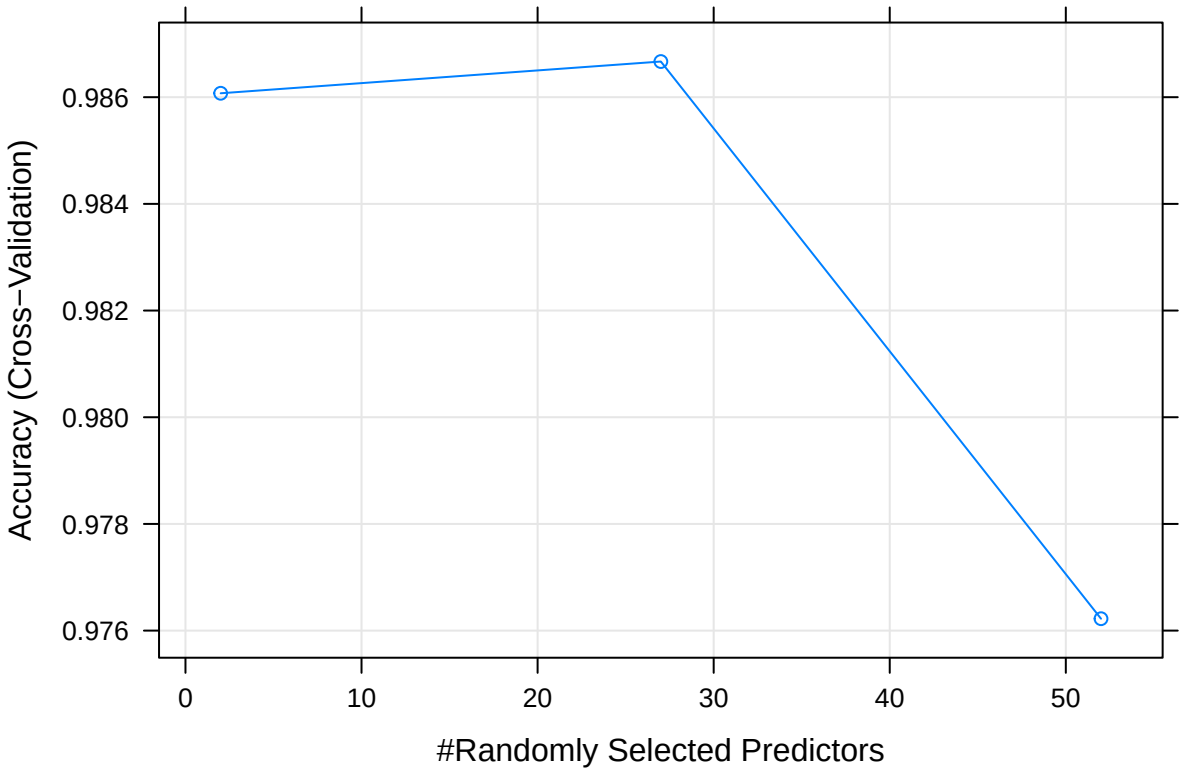

Figure 2: Figure 2: Accuracy of Random Forest Model versus Randomly selected Predictors

```
## roll_forearm          42.693
## pitch_belt            42.540
## magnet_dumbbell_z     41.481
## accel_dumbbell_y      20.550
## magnet_dumbbell_x     17.623
## accel_forearm_x       17.309
## roll_dumbbell         17.194
## magnet_belt_z         15.442
## accel_dumbbell_z      13.954
## total_accel_dumbbell  13.603
## accel_belt_z          13.335
## magnet_forearm_z      13.283
## magnet_belt_y         12.470
## gyros_belt_z          11.223
## magnet_belt_x         10.135
## yaw_arm                9.938
```