Machine Learning Course Project

Sharvaj Phene

10/11/2018

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)

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 are available here: [https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)] The data for this project come 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.

Approach

Our outcome variable is “classe”, a factor variable. For this data set, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions:

Class A - exactly according to the specification Class B - throwing the elbows to the front Class C - lifting the dumbbell only halfway Class D - lowering the dumbbell only halfway Class E - throwing the hips to the front

Two models will be tested using decision tree and random forest. The model with the highest accuracy will be chosen as our final model to predict the “classe” of the testing data set.

Cross Validation

Cross-validation will be performed by subsampling our training data set randomly without replacement into 2 subsamples: subTraining (60% of the original Training data set) and subTesting (remaining 40% of the original Training data set). Our models will be fitted on the subTraining data set, and tested (cross validated) on the subTesting data set. Once the most accurate model is choosen, it will be finally tested on the original testing data set.

Expected Out-of-Sample Error

The expected out of sample error rate will be 1-(cross validation accuracy).

Preparation

```
#load libraries
library(data.table)
library(caret)
```

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

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

```
library(randomForest)
```

```
## 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(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

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

```
#set seed for reproducibility
set.seed(12345)
```

Loading and Partitioning Data for Cross Validation

```
#loading data
training <- read.table(
  paste0(wd, "pml-training.csv"),
  header = TRUE, sep = ",", na.strings = c("NA","#DIV/0!",""))
testing <- read.table(
  paste0(wd, "pml-testing.csv"),
  header = TRUE, sep = ",", na.strings = c("NA","#DIV/0!",""))

#partition training into subTraining(60%) & subTesting(40%) for cross-validation
inTrain <- createDataPartition(y = training$classe, p = 0.6, list = FALSE)
subTraining <- training[inTrain,]
subTesting <- training[-inTrain,]
dim(subTraining)
```

```
## [1] 11776   160
```

```
dim(subTesting)
```

```
## [1] 7846  160
```

Here we can see the dimensions of the subTraining and subTesting sets

Cleaning Data for Model Building

Transformation 1: remove row id variable to prevent interference with ML algorithms

```
subTraining <- subTraining[,-1]
dim(subTraining)
```

```
## [1] 11776   159
```

Transformation 2: removing variables with near zero variance

```
NZV <- nearZeroVar(subTraining, saveMetrics = TRUE)
c <- rownames(NZV[NZV$nzv==FALSE,])
subTraining <- subTraining[,c]
dim(subTraining)
```

```
## [1] 11776   132
```

Transformation 3: remove variables with NAs for over 95% of observations

```
d <- c()
for(i in 1:length(subTraining)) { #iterate for each variable in the training dataset
  if(sum(is.na(subTraining[,i])) / nrow(subTraining) <= .95) { #final list of variables
    d <- c(d, i)
  }
}
subTraining <- subTraining[,d] #retain variables in final list
dim(subTraining)
```

```
## [1] 11776    58
```

Now we must apply the same transformations to our subTesting and testing data sets

```
e <- colnames(subTesting) %in% colnames(subTraining)
subTesting <- subTesting[,e == TRUE]
subTesting1 <- subTesting[,-58] #with classe variable removed
e <- colnames(testing) %in% colnames(subTraining)
testing <- testing[,e == TRUE]
#make sure variable classes are all consistent in the testing data set (merge, then spli
t)
subTesting1$t <- 1
testing$t <- 2
ta <- rbind(testing, subTesting1)
testing <- subset(ta, ta$t == 2)[,-58]
subTesting1 <- subTesting1[,-58]
```

Model Building

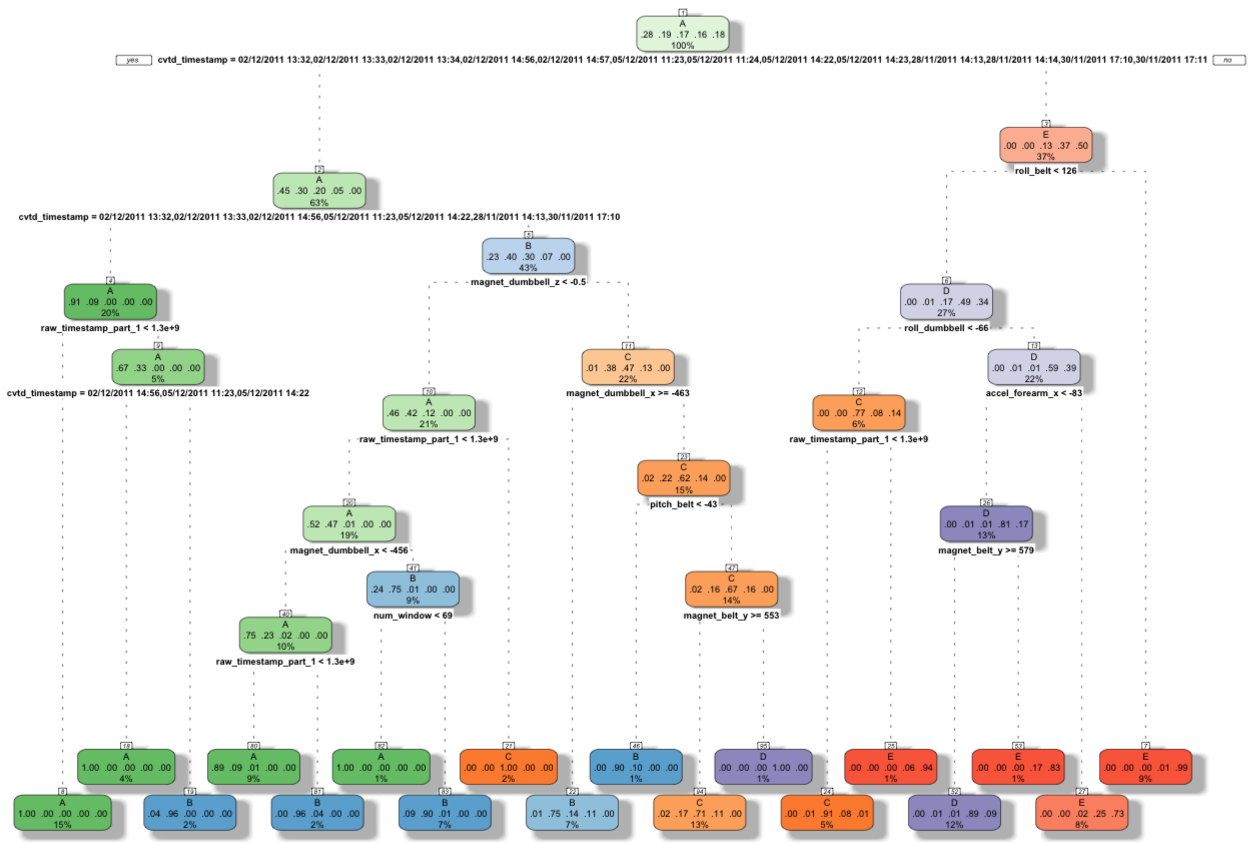
ML Algorithm: Decision Tree

Apply algorithm to subTraining to create predictive model

```
modFit1 <- rpart(classe ~., data = subTraining, method = "class")
```

View created decision tree

```
fancyRpartPlot(modFit1)
```



Rattle 2018-Oct-11 13:12:01 sharvajphene

Cross validate using subTesting data set

```
#use model to predict for subTesting data set
predictions1 <- predict(modFit1, subTesting1, type = "class")
#use confusion matrix to check accuracy of predictions
confusionMatrix(predictions1, subTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2150   60    7    1    0
##          B   61 1260   69   64    0
##          C   21  188 1269  143    4
##          D    0   10   14  857   78
##          E    0    0    9  221 1360
##
## Overall Statistics
##
##                Accuracy : 0.8789
##                  95% CI : (0.8715, 0.8861)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8468
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9633   0.8300   0.9276   0.6664   0.9431
## Specificity            0.9879   0.9693   0.9450   0.9845   0.9641
## Pos Pred Value         0.9693   0.8666   0.7809   0.8936   0.8553
## Neg Pred Value         0.9854   0.9596   0.9841   0.9377   0.9869
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2740   0.1606   0.1617   0.1092   0.1733
## Detection Prevalence   0.2827   0.1853   0.2071   0.1222   0.2027
## Balanced Accuracy      0.9756   0.8997   0.9363   0.8254   0.9536
```

This model’s accuracy is 87.89%, with it’s 95% confidence iterval being (0.8715, 0.8861).

ML Algorithm: Random Forest

Apply algorithm to subTraining to create predictive model

```
modFit2 <- randomForest(classe ~., data = subTraining, method = "class")
```

Cross validate using subTesting data set

```
#use model to predict for subTesting data set
predictions2 <- predict(modFit2, subTesting1, type = "class")
#use confusion matrix to check accuracy of predictions
confusionMatrix(predictions2, subTesting$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    2    0    0    0
##          B    1 1516    1    0    0
##          C    0    0 1366    3    0
##          D    0    0    1 1281    1
##          E    0    0    0    2 1441
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9986          
##                  95% CI : (0.9975, 0.9993)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9982          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9987   0.9985   0.9961   0.9993
## Specificity            0.9996   0.9997   0.9995   0.9997   0.9997
## Pos Pred Value         0.9991   0.9987   0.9978   0.9984   0.9986
## Neg Pred Value         0.9998   0.9997   0.9997   0.9992   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1741   0.1633   0.1837
## Detection Prevalence   0.2846   0.1935   0.1745   0.1635   0.1839
## Balanced Accuracy      0.9996   0.9992   0.9990   0.9979   0.9995
```

This model’s accuracy is 99.86%, with it’s 95% confidence iterval being (0.9975, 0.9993).

Conclusion

This shows that our Random Forest model is more accurate than our Decision Tree model, so we shall proceed with the former.

Final Prediction

Predict classe for testing data set using the RF model (modFit2)

```
finalpred <- predict(modFit2, testing, type = "class")
print(finalpred)
```

```
##  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  C  C  B  A  A  B  C  B  A  C  E  A  B  B  B
## Levels: A B C D E
```