Predicting Weight Lifting Behaviors

A Human Activity Recognition

Pk

Saturday, August 23, 2014

Synopsis:

Human Activity Recognition aims to identify the actions carried out by a person given a set of observations of him/herself and the surrounding environment. Since the 1980s, this research field has captured the attention of several computer science communities due to its strength in providing personalized support for many different applications and its connection to many different fields of study such as medicine, human-computer interaction, or sociology.The approach we propose for the Weight Lifting Exercises dataset is to investigate "how (well)" an activity was performed by the wearer. The "how (well)" investigation has only received little attention so far, even though it potentially provides useful information for a large variety of applications,such as sports training.

One of the most recent, challenging and appealing applications in this framework consists in sensing human body motion 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. 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 study, we present a method to predict what activity a person/subject is performing based on the quantitative measurements that has been created using inertial 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.

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Four classes aligned to common mistakes while Class A corresponds to the specified execution of the exercise.

Methods:

Loading required libraries

```
library(caret)

## Loading required package: lattice
## Loading required package: ggplot2

library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

Downloading, loading and cleaning data

```
# Set working directory
setwd("D:/My docs/Analysis + Cousera/Data Science Specialization/8-
Practical Machine Learning/project")

# Download training data
#download.file(url =
"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv",destfile = "./pml-training.csv")

# Download test data
#download.file(url =
"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv",destfile = "./pml-testing.csv")

#Load training and testing data

training <- read.csv("pml-training.csv")

testing <- read.csv("pml-testing.csv")

### Cleaning Training and Testing data

TraincolumnWithNA <- colSums(is.na(training))  # Identifying columns with
NA and removing them
TrainingNew <- training[!TraincolumnWithNA]    # Removing Na columns

#badColumns <- TraincolumnWithNA >= 19000             # ignoring columns
with majority NA values
#TrainingNew <- training[!badColumns]

#sum(is.na(TrainingNew))

 TrainingNew <- TrainingNew[, c(21:42, 49:51, 61:73, 83:93)]          #
Removing these columns helps with predictions
#TrainingNew <- TrainingNew[, c(83:93)]

#TestcolumnWithNA <- colSums(is.na(testing))   # Identifying columns with
NA and removing them
#TestingNew <- testing[!TestcolumnWithNA]      # Removing Na columns

#TestingNew <- TestingNew[, c(7:60)]            # Removing these columns
helps with predictions
#TestingNew <- TestingNew[, c(21:42, 49:51, 61:73, 83:93)]
```

Exploratory Analysis

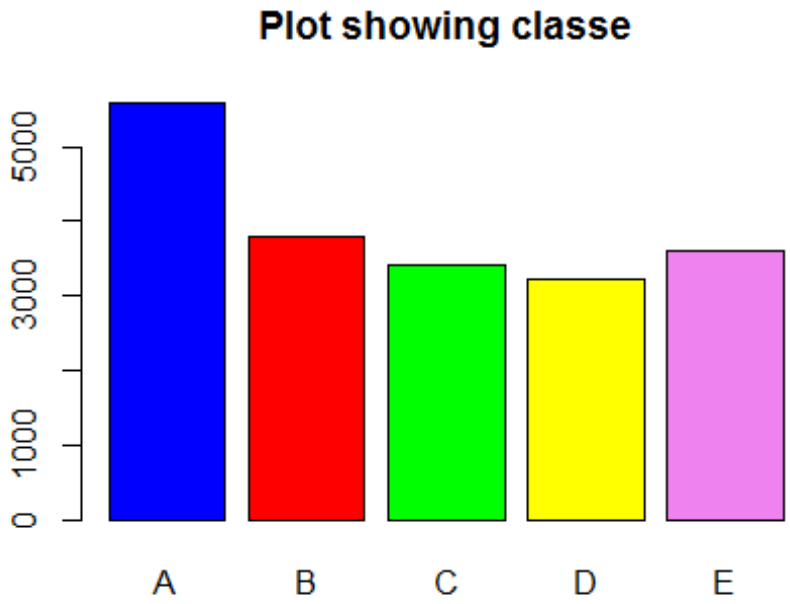
The data was first examined by looking at the names of the variables in the data set and checked to ensure that they were syntactically valid variable names and were not duplicated. Exploratory analysis was performed by examining tables and plots of the observed data. Exploratory analysis was used to (1)gain insight of the data (2)identify missing values, and (3)verify the quality of the data. Summary statistics and a plot after cleaning the data is as shown below:

```
#Summary statistics for the different class grouping

summary(TrainingNew$classe)

##    A    B    C    D    E
## 5580 3797 3422 3216 3607

plot(TrainingNew$classe, col = c("blue", "Red", "Green", "Yellow",
"Violet"), main ="Plot showing classe")
```



Statistical Modelling

We use the random forest package in R to build a classifier for predicting the classe value (activity manner) based on the quantitative measurements recorded from the belt, forearm, arm, and dumbell devices found in our data set.

We first partition our clean data (TrainingNew) into training and testing data.

```
set.seed(100)
Training <- TrainingNew[sample(nrow(TrainingNew), 1500),]  # Only 1,500
cases were used for computational reasons
```

```
partition <- createDataPartition(y = Training$classe, p = 0.7, list =
FALSE)
trainingdata <- Training[partition, ]
testdata <- Training[-partition, ]
```

We now build our model using four-fold cross-validation:

```
model <- train(classe ~ ., data = trainingdata, method = "rf", prox =
TRUE,
               trControl = trainControl(method = "cv", number = 4,
allowParallel = TRUE))
model
```

```
## Random Forest
##
## 1053 samples
##   48 predictors
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 789, 790, 789, 791
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##   2     0.9       0.8    0.04         0.05
##   20    0.8       0.8    0.04         0.04
##   50    0.8       0.8    0.05         0.06
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Results

Calculating In-Sample Accuracy

```
trainPred <- predict(model, trainingdata)
confusionMatrix(trainPred, trainingdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 273   0   0   0   0
##          B   0 209   0   0   0
##          C   0   0 183   0   0
##          D   0   0   0 182   0
##          E   0   0   0   0 206
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.997, 1)
```

```
##     No Information Rate : 0.259
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    1.000    1.000    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    1.000    1.000    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.259    0.198    0.174    0.173    0.196
## Detection Rate          0.259    0.198    0.174    0.173    0.196
## Detection Prevalence    0.259    0.198    0.174    0.173    0.196
## Balanced Accuracy       1.000    1.000    1.000    1.000    1.000
```

We can see from the above statistics that in-sample prediction accuracy is 100% (accuracy = 1)

Calculating Out-Sample Accuracy

```
testPred <- predict(model, testdata)
confusionMatrix(testPred, testdata$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 112   4   3   1   0
##          B   2  80   4   0   9
##          C   0   3  69   5   3
##          D   1   1   2  69   3
##          E   1   1   0   2  72
##
## Overall Statistics
##
##                Accuracy : 0.899
##                  95% CI : (0.868, 0.926)
##     No Information Rate : 0.26
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.873
##  Mcnemar's Test P-Value : 0.0814
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.966    0.899    0.885    0.896    0.828
## Specificity             0.976    0.958    0.970    0.981    0.989
## Pos Pred Value          0.933    0.842    0.862    0.908    0.947
## Neg Pred Value          0.988    0.974    0.975    0.978    0.960
## Prevalence              0.260    0.199    0.174    0.172    0.195
```

```
## Detection Rate          0.251    0.179    0.154    0.154    0.161
## Detection Prevalence    0.268    0.213    0.179    0.170    0.170
## Balanced Accuracy       0.971    0.928    0.927    0.939    0.908
```

Out-sample prediction is 89.9% .

Programming Assignment

Here, we build a prediction algorithm on our model to predict the 20 cases in the test data set:

```
predictClass <- predict(model, testing)
predictClass <- as.character(predictClass)
predictClass
```

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

Conclusion

In this study, our prediction model suggests that it may generalize well on new dataset(new samples) as we observed an overall accuracy of 89.9% for the test data.Prediction accuracy is lower because much computational power was needed to have used all the relevant cases but we had to sample to save time for this assignment. Future work may include all cases and fine tuning the model to enhance the accuracy further then serve to the real life problem solving.