Practical Machine Learning - HAR Dumbbell

Jason Battles

November 13, 2016

An Exercise in Practical Machine Learning using Human Activity Recognition data.

Executive Summary

This analysis describes how a prediction model may be built using acceleromter data to predict with >99% accuracy whether an exercise is correctly being performed.

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit, it is now possible to inexpensively collect a large amount of data about personal activity. 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.

For this study, we will tackle this rarely explored field of "how well" an activity is performed. In this case, 6 young healthy participants were asked to perform 10 repetitions of the **Unilateral Dumbell Biceps Curl** in 5 different classes. Class A was a correctly performed exercise, Class B *incorrectly* threw their elbows forward, Class C *incorrectly* only lifted the dumbell halfway, Class D *incorrectly* only lowered the dumbell halfway, and Class E *incorrectly* threw their hips to the front. The **goal of this study is to predict the manner in which they did the exercise** using only data from accelerometers placed on their belts, forearms, upper arms, and dumbells. In other words, **did the subject perform the biceps curl correctly?** More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Prepare Environment & Data

First, I loaded the necessary environment variables and connect to the raw training data set and test data set. Our training data set consists of accelerometer data and a label identifying the quality of the activity the participant was doing. Our testing data consists of accelerometer data without the identifying label. Our goal is to predict the labels for the test set observations.

```
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(knitr)
library(gbm)
library(ipred)
library(plyr)
library(e1071)

set.seed(2121)
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

```
# load the data and set useless observations to NA as data are loaded
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
dim(training)   # 19,622 observations of 160 variables
```

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

```
dim(testing)    #     20 observations of 160 variables
```

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

I then perform some exploratory data analysis to determine that some of the training data can be eliminated due to large amount of missing and irrelevant data. Those data cannot be used as predictors and should be removed as non-pertinent information to building the prediction models.

```
# remove variables with nearly zero variance
nzv <- nearZeroVar(training)
training <- training[, -nzv]

# remove variables that are almost always NA
mostlyNA <- sapply(training, function(x) mean(is.na(x))) > 0.95
training <- training[, mostlyNA==F]

# remove variables that do not make intuitive sense for prediction (X, user_name, raw_timestamp_part_1,
training <- training[, -(1:6)]
```

Because of the large amount of data available, I decided to further split the training data into a smaller training set (80% of original training data) and a validation set (20% of original training data). I then use the smaller training set to train different models and then I use the new validation set to compute out-of-sample errors after the models are created.

```
inTrain <- createDataPartition(training$classe, p=0.8, list=FALSE)
validation <- training[-inTrain, ]
training <- training[inTrain, ]

# Create new smaller training set and validation set.
dim(training)   # 15,699 observations of 53 variables.
```

```
## [1] 15699    53
```

```
dim(validation) #  3,923 observations of 53 variables.
```

```
## [1] 3923   53
```

Create Models & Measure Accuracy

I would like to control the prediction models in certain ways for experimentation purposes. I intended that all prediction models perform a 5-fold cross valication to save some computational time (default setting in trainControl function is 10). This saved me quite a bit of computational time because I wanted to test so many different prediction models.

```
fitControl1<- trainControl(method = "cv",
                           number = 5,
                           verboseIter = F)
fitControlGBM <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 1)
fitControlRF<- trainControl(method = "cv",
                            number = 5,
                            verboseIter = FALSE)
```

I created 5 different models based on the new 80% smaller training data set and determine which model has the greatest accuracy in predicting outcomes with the 20% validation data set that I created earlier. I discover that **Random Forest (RF) has the greatest accuracy (99.5%)**, followed by Bagged CART (TREEBAG, 98.1%) and Stochastic Gradiant Boosting (GBM, 96.0%). I also was curious about the complexity of each model so I measured the system time spent creating the model. I want to make sure that the model that I create is actually not too cumbersome to implement.

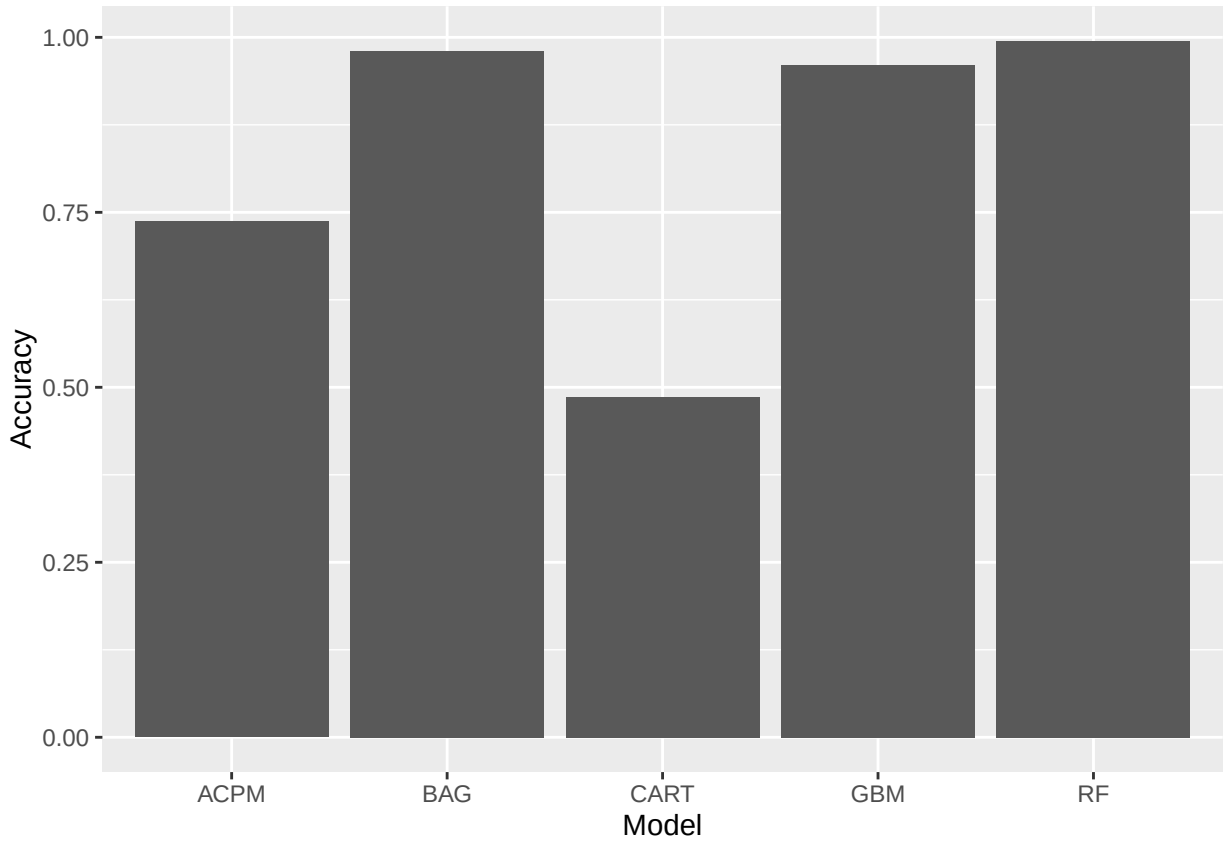
```
ptm <- proc.time()
# CART - Classification & Regression Trees
mdlCART <- train(classe ~ ., data = training,
                 trControl = fitControl1,
                 method = "rpart")
elapsedCART <- proc.time() - ptm

# Stochastic Gradient Boosting
mdlGBM <- train(classe ~ ., data = training,
                 trControl = fitControlGBM,
                 verbose = FALSE,
                 method = "gbm")
elapsedGBM <- proc.time() - ptm

# Random Forest
mdlRF <- train(classe ~ ., data = training,
                 trControl = fitControlRF,
                 method = "rf")
elapsedRF <- proc.time() - ptm

# Bagged CART
mdlBAG <- train(classe ~ ., data = training,
                 trControl = fitControl1,
                 method = "treebag")
elapsedBAG <- proc.time() - ptm

# Adj. Categories Probability Model - Ordinal Data
mdlACPM <- train(classe ~ ., data = training,
                 trControl = fitControl1,
                 method = "vglmAdjCat")
elapsedACPM <- proc.time() - ptm

# Measure complexity by duration of system process time for building each model
TimeCART <- elapsedCART[3]
TimeGBM <- elapsedGBM[3] - TimeCART
TimeRF <- elapsedRF[3] - TimeGBM
```

```
TimeBAG <- elapsedBAG[3] - TimeRF
TimeACPM <- elapsedACPM[3] - TimeBAG

# use modelss to prediction against validation data
# (out-of-sample error  = 1 - Accuracy)
predCART <- predict(mdlCART, newdata = validation)
predGBM <- predict(mdlGBM, newdata = validation)
predRF <- predict(mdlRF, newdata = validation)
predBAG <- predict(mdlBAG, newdata = validation)
predACPM <- predict(mdlACPM, newdata = validation)

# create confusion matrices with prediction outcome characteristics
cmCART <- confusionMatrix(predCART, validation$classe)
cmGBM <- confusionMatrix(predGBM, validation$classe)
cmRF <- confusionMatrix(predRF, validation$classe)
cmBAG <- confusionMatrix(predBAG, validation$classe)
cmACPM <- confusionMatrix(predACPM, validation$classe)

# Random Forest has the highest accuracy (99.46%), followed by Bagged CART (98.06%)
AccuracyResults <- data.frame(
    Model = c('CART', 'GBM', 'RF', 'BAG', 'ACPM'),
    Accuracy = rbind(cmCART$overall[1], cmGBM$overall[1], cmRF$overall[1], cmBAG$overall[1], cmACPM$ove
    Duration = rbind(TimeCART, TimeGBM, TimeRF,TimeBAG,TimeACPM)
)
print(AccuracyResults)
```

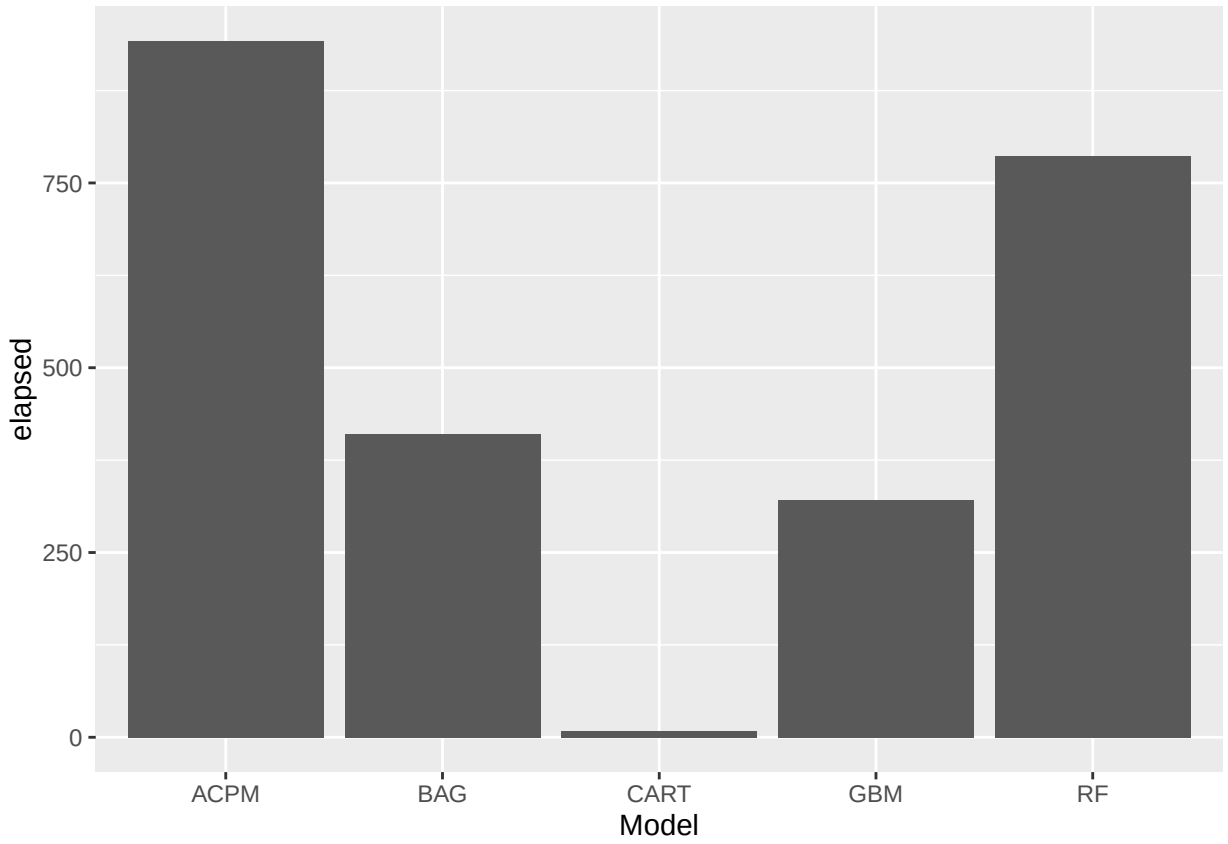
```
##          Model  Accuracy elapsed
## TimeCART  CART 0.4861076   8.316
## TimeGBM    GBM 0.9604894 320.816
## TimeRF      RF 0.9946470 786.556
## TimeBAG    BAG 0.9806271 410.484
## TimeACPM  ACPM 0.7369360 941.937
```

Here also are two bar plots to represent (1) Accuracy of the different models and (2) Complexity of the models as measured by system processing time

```
acc.barplot <- ggplot(AccuracyResults, aes(x=Model, y=Accuracy)) + geom_bar(stat="identity")
acc.barplot
```

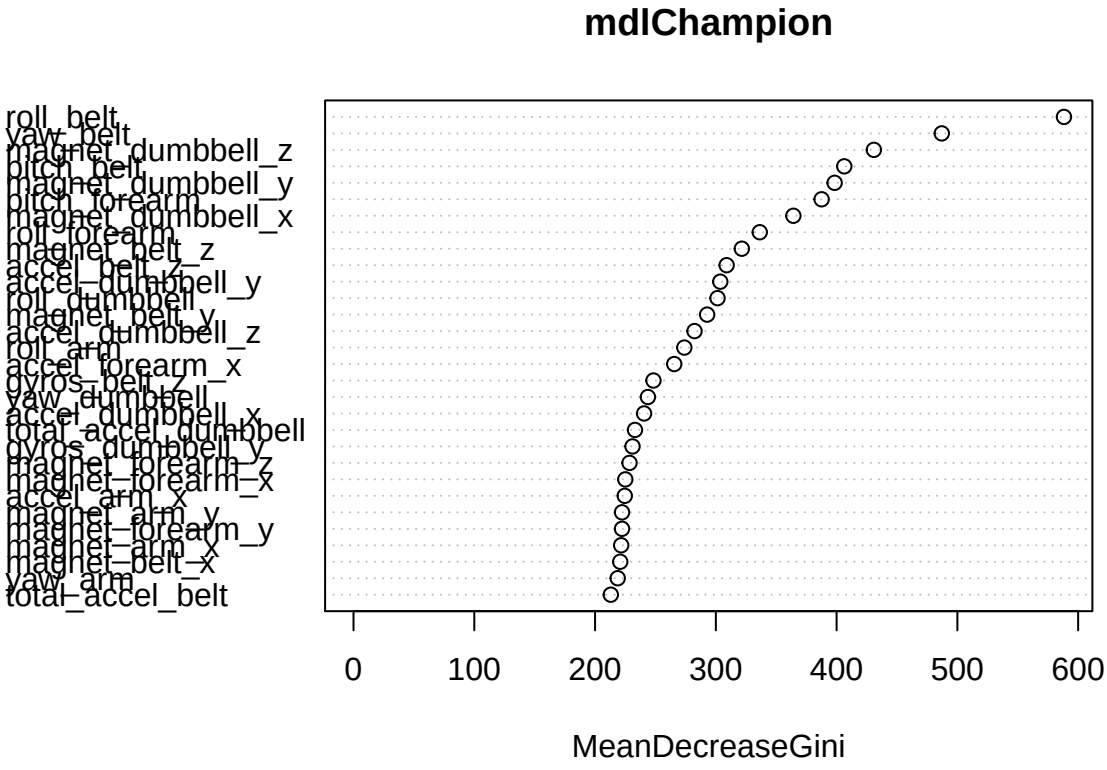



```
dur.barplot <- ggplot(AccuracyResults, aes(x=Model, y=elapsed)) + geom_bar(stat="identity")
dur.barplot
```



The Random Forests algorithm chooses a subset of predictors at each split and decorrelates the trees. This leads to high accuracy, although this algorithm is sometimes difficult to interpret and sometimes computationally inefficient. With such a high accuracy and low error rate in my computations (out-of-sample error = 0.05%), **I declared my Random Forest prediction model as the champion model**. Before I continued with the final predictions against the testing data set, I was curious to discover which variables were in the majority in their contributions to the model. The top 3 contributor variables were **roll_belt**, **yaw_belt**, and **magnet_dumbbell_z**.

```
mdlChampion <- mdlRF$finalModel
varImpPlot(mdlChampion)
```



Predict Outcomes

Since I have created a prediction model and validated it with a high level of accuracy, I feel reasonably confident in predicing the outcomes on 20 observations in the testing data. These predictions are as follows.

```
predTest <- predict(mdlRF, newdata=testing)

ValidationTestResults <- data.frame(
    problem_id=testing$problem_id,
    predicted=predTest
)
summary(ValidationTestResults$predicted)
```

```
## A B C D E
## 7 8 1 1 3
```

```
ValidationTestResults
```

```
##    problem_id predicted
## 1           1         B
## 2           2         A
## 3           3         B
## 4           4         A
## 5           5         A
## 6           6         E
## 7           7         D
## 8           8         B
```

```
## 9            9         A
## 10          10         A
## 11          11         B
## 12          12         C
## 13          13         B
## 14          14         A
## 15          15         E
## 16          16         E
## 17          17         A
## 18          18         B
## 19          19         B
## 20          20         B
```

Conclusions

1. Out of 20 exercise observations in the testing data set, only **7 were correctly performed (Class A)**.
2. The **most common incorrectly performed exercise** was **Class B** and is characterized by swinging the elbows forward.
3. The **Random Forest** model was the **Most Accurate** model with a **99.5% degree of accuracy**. The Random Forest algorithm outperformed the 4 other models that were attempted with this data.
4. The **Adjusted Categories Probability Model for Ordinal Data (ACPM)** was the most computationally complex model to create with over **911 seconds** spent during its training process.
5. **Random Forest** was the 2nd most computationally complex model. It took **787 seconds** to train this model