Predicting barbell lift style

LMarshall

March 11, 2017

Executive Summary

We are looking at the Weight Lifting Exercises Dataset made available by E. Velloso et al (http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har)). The goal is to build a model based on a training data set that will accurately predict outcomes in a test set. The data consists of measured body movements corresponding to 5 different barbell lifting styles (A,B,C,D, and E), and we want to predict the style based on the measurements given.

The training set consists of 19622 observations and 160 variables. I used 59 variables and ultimately selected a random forest model that yielded an accuracy of .9994 that was able to accurately predict all 20 outcomes in the test set.

Building the Model

To build the model, I load and explore the data sets and read the documentation available via the link given in the summary above. My exploration of the data's dimensions, summary, structure, etc is not shown below, given the report's length limits. What is shown are the data cleaning and variable-removal choices I make.

I choose to get rid of the NA-heavy columns because there isn't enough information in the documentation (specifically about the new-window variable) to determine a good way to impute values there.

```
library(caret)
```

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

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

```
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
trainURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.c
sv"
download.file(trainURL, destfile="trainURL.csv")
testURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.cs
v"
download.file(testURL, destfile="testURL.csv")

#I'm getting pesky "#DIV/0!" instances in the data. Will set those to NA's
training <- read.csv('trainURL.csv', na.strings=c("NA","#DIV/0!", ""))
testing <- read.csv('testURL.csv', na.strings=c("NA","#DIV/0!", ""))

#need to remove the X variable, since it appears to just be an index
training2 <- training[,-1]
testing2 <- testing[,-1]

#get rid of cols that only have 1 level
kill_list <- as.vector(sapply(training2, class))
kill_list_ <- which(kill_list=="logical")
training3 <- training2[,-kill_list_]
testing3 <- testing2[,-kill_list_]

#get rid of cols that are mostly NA's
training4 <- training3[,colMeans(is.na(training3)) < .8]
testing4 <- testing3[,colMeans(is.na(testing3)) < .8]
```

Cross-validation

I split the training set into 2 subsets (75% and 25% of the data). I use the larger one for training and the smaller one to cross-validate the models I am testing.

```
#separate training data into 2 subsets
inTrain <- createDataPartition(training4$classe, p=.75, list=FALSE)
firsthalf <- training4[inTrain,]
secondhalf <- training4[-inTrain,]
```

First model fit: Regression tree

I use rpart to fit my first model attempt

```
mod1 <- train(classe~., data=firsthalf, method='rpart')
p <- predict(mod1, newdata=secondhalf)
cm <- confusionMatrix(p, secondhalf$classe)
cm$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   4.584013e-01   3.035658e-01   4.443821e-01   4.724701e-01   2.844617e-01
## AccuracyPValue  McnemarPValue
##  5.690794e-147            NaN
```

The accuracy is quite low, so I look for another modeling option.

Second model fit: Random forest

I use random forest next, as it has a reputation for high accuracy.

```
mod3_rf <- randomForest(classe~., data=firsthalf)
p2 <- predict(mod3_rf, newdata=secondhalf[,-59])
cm2 <- confusionMatrix(p2, secondhalf$classe)
cm2$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9997961      0.9997421      0.9988644      0.9999948      0.2844617
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

The accuracy here looks really good .9994. The error rate is 1-.9994/100, so I expect about a **~1% out of sample error**.

Using the model

When I attempt to use my winning model on the test set, I run into errors because the levels in a couple of the factor variables don't match. So I need to match those levels before proceding. Then I run the prediction.

```
levels(testing4$cvtd_timestamp) <- levels(training4$cvtd_timestamp)
levels(testing4$new_window) <- levels(training4$new_window)
testing5<-testing4[,-59]
ptest <- predict(mod3_rf, newdata=testing5)

#prediction for the 20 unknowns in the test set
ptest
```

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

Conclusion

Running my prediction results through the quiz shows me all of my predictions are accurate. Since the expected out of sample error is 1% and I only made predictions for 20 cases, that is the outcome I would have expected.