PML - Course Project

December 26, 2015

Overview

The goal of this porject is to use the Weight Lifting Exercise Dataset from to build a model that can predict if participants were performing a curl correctly or performing an incorrect motion in 1 of 5 ways. Information about the data can be found at http://groupware.les.inf.puc-rio.br/har. This write-up will discuss perparing the data for modeling, fitting the model, and predicting the 20 test cases.

Data Preparation

The initial data is in a raw form with some blanks and error codes. We set these to NA in our read statement.

```
library(caret)
```

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

```
download.file(url='https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv',
              destfile='./PML_train.csv')

#Read in data setting empty and error values to NA.
WL <- read.csv('./PML_train.csv',na.strings = c("NA","#DIV/0!",""),as.is = TRUE)
WL <- data.frame(WL)
```

Create a training and test set to estimate out of sample error.

```
inTrain <- createDataPartition(y = WL$classe, p=.75, list = F)
WLtrain <- WL[inTrain,]
WLtest <- WL[-inTrain,]
```

The predictors are to be measurments from various sensors worn by the participants. Thus for predition purposes I'll ignore the first 6 columns which are participant name, timestamps, and other experiment variables.

```
exp_vars <- grep("X|user_name|raw_timestamp_part_1|raw_timestamp_part_2|cvtd_timestamp|num_window",name
WLtrain <- WLtrain[,-exp_vars]
```

A number of the predictors still have many missing values. There are only 221 complete observations all of the predictors. I've fit a Gradient Boosting Machine with the complete dataset with missing values, but accuracy was only 49.93% (not shown). Far greater accuracy can be achieved by removing the variables with missing values.

```
#clean variables with missing values
null_count <- sapply(WLtrain, function(x) sum(is.na(x)))
keep <- null_count==0
WLtrain <- WLtrain[,keep]
```

The prediction target is the classe variable. We'll make this a factor and look at a quick frequency and plot it by two of the most important varaibles.

```
WL$classe <- factor(WL$classe)
table(WL$classe)
```

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

```
library(ggplot2)
ggplot(aes(x=roll_belt,y=magnet_dumbbell_y,colour=classe), data=WLtrain) + geom_point() + coord_cartesia
```

Modeling

We'll fit a random forrest to the model. We'll use 5 fold cross validation in the train control to help mitigate potential overfitting.

```
Control1 <- trainControl(method = "cv", number = 5)
modFit1 <- train(classe ~ ., data=WLtrain, method ="rf", importance = T, trControl = Control1)
```

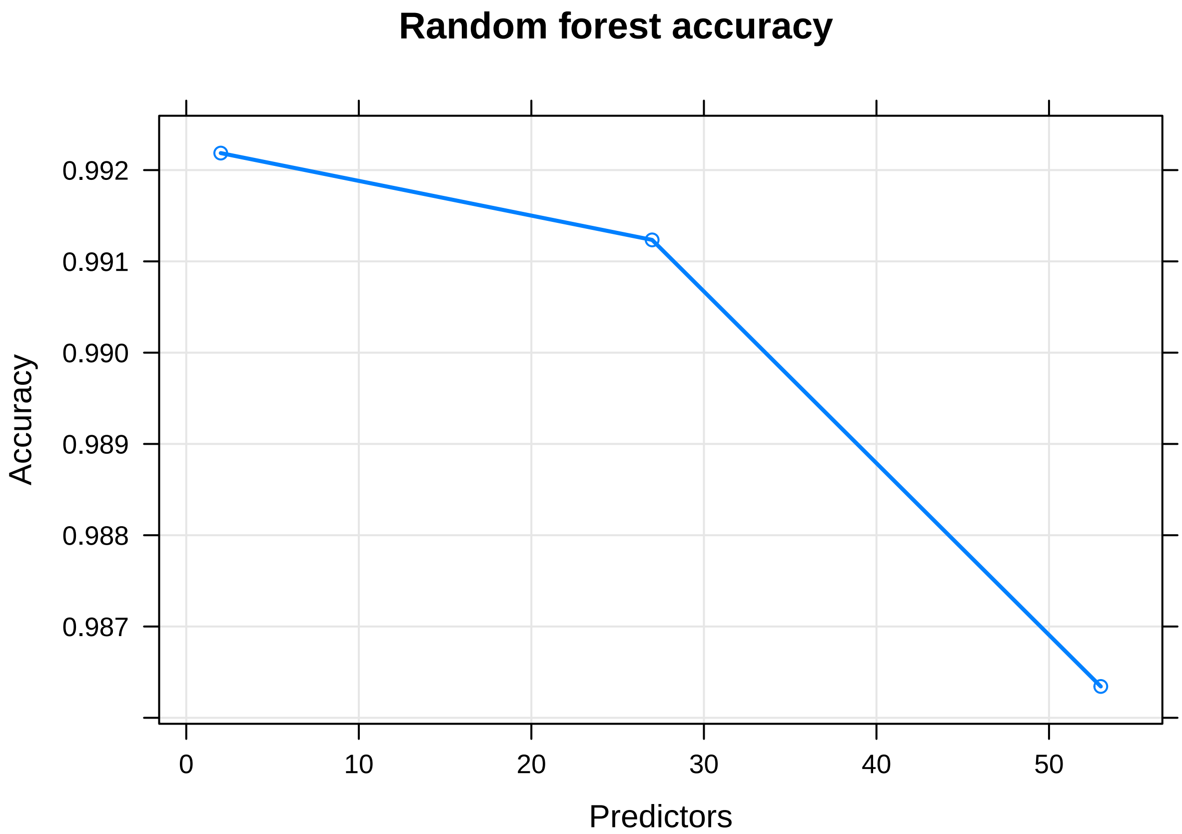
```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```r
modFit1
```

```
## Random Forest
##
## 14718 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11774, 11775, 11773, 11775, 11775
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9921864  0.9901157  0.001293926  0.001636516
##   27    0.9912354  0.9889124  0.001652847  0.002091711
##   53    0.9863434  0.9827231  0.002307971  0.002922301
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

We can examine the selection process as predictors were added.

```r
plot(modFit1, log = "y", lwd = 2, main = "Random forest accuracy", xlab = "Predictors",  ylab = "Accurac
```



The final model utilized 27 predictors. “Roll belt” and “magnet dumbbell y” were the most important predictors.

```r
varImp(modFit1)
```

```
## rf variable importance
##
##   variables are sorted by maximum importance across the classes
##   only 20 most important variables shown (out of 53)
##
##                       A      B     C     D     E
## roll_belt         81.81 100.00 95.00 93.74 75.81
## pitch_belt        80.71  94.49 76.39 79.93 81.12
## yaw_belt          89.22  93.54 85.92 91.85 65.96
## roll_arm          66.94  90.30 86.87 79.99 73.27
## magnet_dumbbell_z 83.60  85.11 89.72 77.81 74.15
## magnet_dumbbell_y 70.44  77.32 88.77 75.25 68.99
## accel_dumbbell_y  65.16  79.25 73.60 80.37 72.93
## pitch_forearm     66.15  80.15 77.80 77.10 70.06
## yaw_arm           64.33  78.03 67.43 71.69 66.66
## accel_dumbbell_z  65.99  77.69 71.29 69.28 69.74
## gyros_arm_y       57.60  77.32 63.47 68.36 59.00
## magnet_dumbbell_x 60.11  70.38 77.06 62.69 58.44
## gyros_forearm_y   55.56  76.87 64.26 70.14 61.72
## gyros_belt_z      63.77  76.33 70.89 73.08 69.72
## accel_belt_z      64.26  74.44 70.41 64.64 61.59
## magnet_forearm_z  60.18  74.41 68.65 72.02 66.32
## gyros_dumbbell_y  62.55  73.05 71.85 70.58 61.05
## roll_dumbbell     60.70  64.54 72.90 67.45 63.73
## gyros_dumbbell_x  61.35  72.33 68.00 60.77 59.18
## gyros_dumbbell_z  54.64  71.82 60.71 60.58 61.35
```

Out of Sample Error Prediction

To estimate the out of sample error we apply the prediction to the test set.

```r
test.Predict = predict(modFit1, WLtest)
confusionMatrix(test.Predict,WLtest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    5    0    0    0
##          B    0  937    4    0    0
##          C    0    7  850   10    0
##          D    0    0    1  794    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.992, 0.9964)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.993
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9874   0.9942   0.9876   1.0000
## Specificity            0.9986   0.9990   0.9958   0.9998   1.0000
## Pos Pred Value         0.9964   0.9957   0.9804   0.9987   1.0000
## Neg Pred Value         1.0000   0.9970   0.9988   0.9976   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1911   0.1733   0.1619   0.1837
## Detection Prevalence   0.2855   0.1919   0.1768   0.1621   0.1837
## Balanced Accuracy      0.9993   0.9932   0.9950   0.9937   1.0000
```

The out of sample error prediction is 1-accuracy, which has an estimate of 0.86% with a 95% CI of (0.62%, 1.16%).