Machine Learning Project

Edward Bruggemann

June 13, 2016

Summary

The data for this project comprises accelerometer output from the belt, forearm, arm, and dumbell of 6 participants. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The goal is to predict the manner in which the participants did the exercise from the accelerometer output. A training dataset and a testing dataset were provided. The caret package was used to build and test models, and to make predictions. A random forest model successfully predicted the testing classifications.

```
## load caret
library(caret)
```

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

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

Datasets

I downloaded the datasets using the URL's provided in the instructions.

```
url1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
train1 <- read.csv(url1)
test1 <- read.csv(url2)
```

The training dataset comprises 19,622 rows x 160 columns and the testing dataset comprises 20 rows by 160 columns. There is only one column name difference. Column "classe" in the training data contains the known classifications as a factor of five levels: A, B, C, D, and E. In the testing dataset this column is replaced with "problem_id", numbered 1 - 20, which corresponds to the numbered quiz questions for submission.

```
names(train1)[!names(train1)==names(test1)]
```

```
## [1] "classe"
```

```
names(test1)[!names(train1)==names(test1)]
```

```
## [1] "problem_id"
```

Dataset Cleanup

Both datasets contain many columns that are entirely or almost entirely NA; all other columns are complete. I included for further analysis only columns that were complete in both datasets. Train2 and Test2 both contain 60 columns, of which 59 are potential predictors and 1 is the response variable.

```
keepcols <- colSums(is.na(train1))==0 & colSums(is.na(test1))==0
train2 <- train1[, keepcols]
test2 <- test1[, keepcols]
```

Inspection of column names and data revealed that columns 1-7 could be excluded as potential predictors. Train3 and Test3 both 53 columns, of which 52 are potentail predictors and 1 is the response variable.

```
names(train2)[1:7]
```

```
## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
## [7] "num_window"
```

```
train3 <- train2[, -(1:7)]
test3 <- test2[, -(1:7)]
```

Dataset Processing

A quick search for variables with low variance or no variance revealed none.

```
low.var <- nearZeroVar(train3[, -53], saveMetrics=TRUE)
sum(low.var$zeroVar)
```

```
## [1] 0
```

```
sum(low.var$nzv)
```

```
## [1] 0
```

A quick search for variables with high correlation revealed a handful, which I excluded from further analysis. The cutoff of 0.80 is arbitrary. Train4 and Test4 both contain 40 columns, of which 39 are potential predictors and 1 is the response variable.

```
cormat <- cor(train3[, -53])
highcorr <- findCorrelation(cormat, cutoff=0.80)
train4 <- train3[,-highcorr]
test4 <- test3[,-highcorr]
```

Cross Validation

I partitioned the training data into train4A for training the model and train4B for testing and estimating out-of-sample accuracy before applying the model to the testing dataset.

```
set.seed(4674833)
partindex <- createDataPartition(train4$classe, p=0.80, list=FALSE)
train4A <- train4[partindex,]
train4B <- train4[-partindex,]
```

Building the Model

I selected a random forest model because in the lectures it was asserted several times that this method is often the top performer. I used default parameter settings in all cases, except for trainControl(method="cv"), which appeared to be faster than method="boot" on my machine.

```
## very slow
t1 <- Sys.time()
set.seed(9892225)
model <- train(classe~., data=train4A, method="rf",
               trControl=trainControl(method = "cv"))
```

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

```
t2 <- Sys.time()
t2-t1
```

```
## Time difference of 24.50457 mins
```

Predictions from the Model

I used this model to predict on Train4B. Out-of-sample accuracy was 0.9934. Good enough to proceed to the testing dataset.

```
predictB <- predict(model, newdata=train4B)
conmatB <- confusionMatrix(predictB, train4B$classe)
print(conmatB)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    7    0    0    0
##          B    0  747    2    0    0
##          C    0    5  677    4    1
##          D    0    0    5  639    2
##          E    0    0    0    0  718
##
## Overall Statistics
##
##                Accuracy : 0.9934
##                  95% CI : (0.9903, 0.9957)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9916
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9842   0.9898   0.9938   0.9958
## Specificity            0.9975   0.9994   0.9969   0.9979   1.0000
## Pos Pred Value         0.9938   0.9973   0.9854   0.9892   1.0000
## Neg Pred Value         1.0000   0.9962   0.9978   0.9988   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1904   0.1726   0.1629   0.1830
## Detection Prevalence   0.2863   0.1909   0.1751   0.1647   0.1830
## Balanced Accuracy      0.9988   0.9918   0.9933   0.9958   0.9979
```

I used this model to predict the testing set and submitted the predictions for grading. The result was 100% accurate.

```
predict.test <- predict(model, newdata=test4)
print(predict.test)
```

```
##  [1] 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
```