Practical Machine Learning Course Project

E.T. Moseley

January 28, 2017

Executive Summary

how you built your model how you used cross validation what you think the expected out of sample error is why you made the choices you did

In the aforementioned study, six participants participated in a dumbell lifting exercise five different ways.

The five ways, as described in the study, were "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).

By processing data gathered from accelerometers on the belt, forearm, arm, and dumbell of the participants in a machine learning algorithm, the question is can the appropriate activity quality (class A-E) be predicted?

```
require("caret")
```

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

```
require("rpart")
```

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

```
require("nnet")
```

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

```
#require("rattle")#Causes my mac to crash
require("e1071")
```

```
## Loading required package: e1071
```

```
require("gbm")
```

```
## Loading required package: gbm
## Loading required package: survival
##
## Attaching package: 'survival'
## The following object is masked from 'package:caret':
##
##     cluster
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
```

```
train <- read.csv("/Users/Edward/Desktop/PracticalMachineLearning/pml-training.csv", header = TRUE, stri
```

We will view the dimension's of the training set

```r
dim(train)
```

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

We see there are 19622 observations of 160 variables.

```r
head(colnames(train))
```

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

Data Preprocessing

We will be predicting the `classe`, but let us remove columns `X`, `user_name`, `raw_timestamp_part_1`, `raw_timestamp_part_2` and `cvtd_timestamp` as they should not be predictive in our model.

```r
train$X <- NULL
train$user_name <- NULL
train$raw_timestamp_part_1 <- NULL
train$raw_timestamp_part_2 <- NULL
train$cvtd_timestamp <- NULL
```

Let's partition data 80%-20%

```r
set.seed(123)
trainSet <- createDataPartition(y=train$classe, p=0.8, list=FALSE)
trainOne <- train[trainSet, ]
testOne <- train[-trainSet, ]
```

We can remove variables with near-zero variance, which are unlikely to add predictive power to our model.

```r
lowVar <- nearZeroVar(trainOne, saveMetrics=TRUE)
head(lowVar)
```

```
##                  freqRatio percentUnique zeroVar   nzv
## new_window       47.603715    0.01273966   FALSE  TRUE
## num_window        1.064516    5.45894643   FALSE FALSE
## roll_belt         1.124646    7.56099115   FALSE FALSE
## pitch_belt        1.108844   11.14720683   FALSE FALSE
## yaw_belt          1.099515   11.88610740   FALSE FALSE
## total_accel_belt  1.079975    0.18472514   FALSE FALSE
```

Now we will remove these low-variability data from our sets:

```r
trainOne <- trainOne[,!(lowVar$nzv)]
testOne <- testOne[,!(lowVar$nzv)]
dim(trainOne)
```

```
## [1] 15699    99
```

Furthermore, throughout this analysis we will be ignoring NA values with `na.action = na.pass`.

Modeling

Now we will generate an rpart model and evaluate it:

CART (Classification and Regression Tree) Model

```r
attach(trainOne)
rpartModel <- train(classe ~ ., data = trainOne, method="rpart", na.action = na.pass)
rpartModel
```

```
## CART
##
## 323 samples
##  98 predictor
##   5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 15699, 15699, 15699, 15699, 15699, 15699, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03871829  0.5587318  0.43772113
##   0.05957573  0.4022082  0.18484243
##   0.11544281  0.3272564  0.06492671
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03871829.
```

Let's apply this model to our test set:

```r
confusionMatrix(predict(rpartModel, newdata=testOne, na.action = na.pass), testOne$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1011  308  304  283  107
##          B   16  262   26  132   87
##          C   85  189  354  228  203
##          D    0    0    0    0    0
##          E    4    0    0    0  324
##
## Overall Statistics
##
##                Accuracy : 0.4973
##                  95% CI : (0.4816, 0.5131)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3436
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9059  0.34519  0.51754   0.0000  0.44938
## Specificity            0.6430  0.91751  0.78234   1.0000  0.99875
## Pos Pred Value         0.5022  0.50096  0.33428      NaN  0.98780
```

```
## Neg Pred Value         0.9450  0.85382  0.88478   0.8361  0.88957
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2577  0.06679  0.09024   0.0000  0.08259
## Detection Prevalence   0.5131  0.13332  0.26995   0.0000  0.08361
## Balanced Accuracy      0.7745  0.63135  0.64994   0.5000  0.72406
```

Out-of-sample Error

It is helpful to view the accuracy of the model:

```
confusionMatrix(predict(rpartModel, newdata=testOne, na.action = na.pass), testOne$classe)$overall["Accu
```

```
##  Accuracy
## 0.4973235
```

Unfortunately, the accuracy is relatively low.