Practical Machine Learning Course Project

Thomson Kneeland

June 14, 2016

Executive Summary

Using the Weight Lifting Exercises Dataset, we examined the execution of the Unilateral Dumbbell Biceps Curl with 6 participants performing 10 repetitions in 5 different fashions: 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). (See http://groupware.les.inf.puc-rio.br/har for more details.)

Our goal is to create a viable machine learning algorithm to identify the class of future observations based on the data presented. The first model attempted was a decision tree model which yielded only 49% accuracy and was dismissed. A second model used gradient boosting and yielded a 96.3% accuracy (out of sample error of 3.6%), faring quite well for prediction purposes. However, a third model using random forests fared even better with an accuracy of 99.3% and an out-of-bag (OOB) estimate of error of .43%.

Data Preparation and Cleaning

First we load our training and test data along with any packages needed for data processing. We also account for missing variable entries by providing NA values.

```
data.train <- read.csv("pml-training.csv", stringsAsFactors = FALSE, header=TRUE,
                       na.strings=c("","#DIV/0!"))
data.test <- read.csv("pml-testing.csv", stringsAsFactors = FALSE, header=TRUE,
                      na.strings=c("","#DIV/0!"))
library(caret)
library(dplyr)
library(rpart)
library(rattle)
library(randomForest)
```

The first seven variable columns do not feature data that we need for processing.

```
data.train <- select(data.train, -(1:7))
data.test <- select(data.test, -(1:7))
```

Variables with nearly zero variance will not help with our predictions, so we remove those.

```
nzv <- nearZeroVar(data.train)
data.train <- data.train[, -nzv]
```

Some columns feature all NA values, so those will be removed as well.

```
var.comp <- names(data.train[,colSums(is.na(data.train)) == 0])
data.train <- data.train[, var.comp]
```

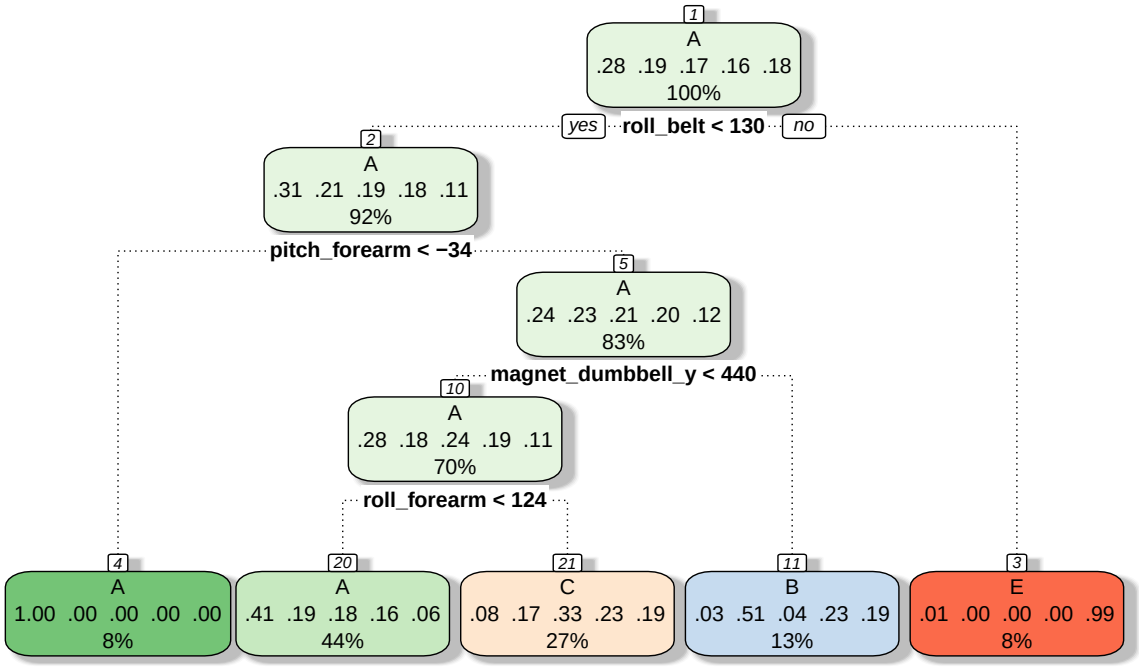
The final dataset now consists of 52 variables and our outcome (classe). Finally, we will also subset the training data for cross validation purposes in our decision tree model with a 70/30% allocation.

```
set.seed(412)
sub <- createDataPartition(data.train$classe, p = .7,list=FALSE)
data.train.1 <- data.train[sub,]
data.train.2 <- data.train[-sub,]
```

Decision Tree Model

We build a decision tree model on the first partition (70%) of our training data.

```
model.tree <- train(classe ~ .,method="rpart",data=data.train.1)
fancyRpartPlot(model.tree$finalModel,sub="")
```



Using this model, we will predict and verify the outcomes of the second partition (30%) of our training data for cross validation.

```
tree.predict <- predict(model.tree,newdata=data.train.2)
confusionMatrix(tree.predict, data.train.2$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1524  476  490  471  182
##          B   32  379   30  157  145
##          C  115  284  506  336  273
##          D    0    0    0    0    0
##          E    3    0    0    0  482
```

```
##
## Overall Statistics
##
##                Accuracy : 0.4912
##                  95% CI : (0.4784, 0.5041)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3338
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9104   0.3327  0.49318   0.0000  0.44547
## Specificity            0.6155   0.9233  0.79255   1.0000  0.99938
## Pos Pred Value         0.4849   0.5101  0.33421      NaN  0.99381
## Neg Pred Value         0.9453   0.8522  0.88103   0.8362  0.88889
## Prevalence             0.2845   0.1935  0.17434   0.1638  0.18386
## Detection Rate         0.2590   0.0644  0.08598   0.0000  0.08190
## Detection Prevalence   0.5341   0.1263  0.25726   0.0000  0.08241
## Balanced Accuracy      0.7630   0.6280  0.64286   0.5000  0.72242
```

The decision tree model on the partitioned data yields a prediction accuracy of 49.1%, not a good fit; the Kappa statistic of .33 is unimpressive as a measure of matching the outcome. We should seek another model with greater accuracy.

Gradient Boosting Model

We next fit a gradient boosting model on the partitioned data. A preliminary run gives us best fit parameters of 150 trees and more, so we have added these to our model to save extensive calculation time.

```
model.gbm <- train(classe ~ .,method="gbm",data=data.train.1,verbose=FALSE)
gbm.predict2 <- predict(model.gbm,newdata=data.train.2,n.trees=150,interaction.depth = 3,shrinkage = 0.
                        n.minobsinnode = 10)
gbm.predict <- predict(model.gbm,newdata=data.train.2)
confusionMatrix(gbm.predict, data.train.2$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1651   37    0    0    5
##          B    8 1071   32    0   14
##          C   11   29  981   26    6
##          D    2    2   10  931   22
##          E    2    0    3    7 1035
##
## Overall Statistics
##
##                Accuracy : 0.9633
##                  95% CI : (0.9582, 0.968)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9536
##  Mcnemar's Test P-Value : 4.068e-10
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9863   0.9403   0.9561   0.9658   0.9566
## Specificity            0.9900   0.9886   0.9852   0.9927   0.9975
## Pos Pred Value         0.9752   0.9520   0.9316   0.9628   0.9885
## Neg Pred Value         0.9945   0.9857   0.9907   0.9933   0.9903
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2805   0.1820   0.1667   0.1582   0.1759
## Detection Prevalence   0.2877   0.1912   0.1789   0.1643   0.1779
## Balanced Accuracy      0.9881   0.9645   0.9707   0.9792   0.9770
```

The gbm model fares much better, with a 96.3% accuracy rate in predicting our out of sample data with a Kappa statistic of .954. This does seem like a great model fit, but we will explore a random forest model to see if it will outperform.

Random Forest Model

The random forest function cross validates internally, so we will use a 3-fold cross validation on the full training data. The model will grow 200 trees, enough trees for accuracy and insuring input rows are predicted multiple times, while lowering processing time.

```
model.rf <- train(classe ~ .,method="rf",ntree=200,data=data.train,
                  trControl=trainControl(method="cv",number=3),keep.forest=TRUE)
model.rf
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 13080, 13083, 13081
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9927633  0.9908454
##   27    0.9931709  0.9913617
##   52    0.9857300  0.9819487
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

All three models produce stellar results > 98.5% accuracy with very high Kappa statistic values > .98.

```
model.rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 200, mtry = param$mtry, keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.43%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5576    2    1    0    1 0.0007168459
## B   20 3771    5    1    0 0.0068475112
## C    0   10 3402   10    0 0.0058445354
## D    0    0   21 3192    3 0.0074626866
## E    0    0    4    7 3596 0.0030496257
```

This best fit model features an accuracy of 99.3% using 27 variables, far outperforming the decision tree model and being a perfect candidate for future prediction. The out-of-bag (OOB) error estimate for this model is .43%, supporting its strength with a low predictive error. Accordingly, the classification error is minimal and less than 1% for all outcomes. The OOB error estimate removes the need for an out of sample error rate, since 1/3 of the training data set was used as "out of bag samples". We can expect this error rate to remain similar in predicting future outcomes.

Conclusions and Test Set Predictions

The Decision Tree model performed inadequately, only predicting 49.1% of outcomes. The GBM and random forest models performed quite well with an accuracy of 96.3% and 99.3% respectively. The random forest model was selected as the best fit and we will use it on our test data. Accordingly, we predict the following results for classe with the random forest model:

```
predict(model.rf,data.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
```