Machine Learning HAR

Ken Duffau

October 8, 2016

Executive Summary

The Human Activity Recognition (HAR) Weightlifting Exercise dataset contains 19622 observations, with 160 variables, of six human subjects performing standard curls. The objective of the original study was to determine if the researchers could identify measurements that determined whether an exercise iteration was performed correctly. The researchers developed five classes from the measurements with Class A notated as a correctly performed iteration. The calculated measures were derived from features on the Euler angles (roll, pitch and yaw), as well as the raw accelerometer, gyroscope and magnetometer readings. For the Euler angles of each of the four sensors they calculated eight features: mean, variance, standard deviation, max, min, amplitude, kurtosis and skewness. We developed several machine learning models to predict how exercises would be performed in a test set of 20 observations, and after significant cleaning and evaluating seven separate models, we determined that R's Caret package "rf" method with 5-fold cross-validation returned the highest accuracy (99.8%). It was noticeable however that the standalone randomForest package was much quicker in fitting the model and the accuracy was "close enough (99.6%)" that we found the latter package to be equally acceptable.

Analysis Approach

Data Preparation

We started with simple exploration of the dataset and quickly identified many variables with a predominance of NA values. Despite identifying only 406 complete cases we determined that we should try to maximize the number of training observations. Instead of filtering on complete cases, we elected to remove all variables that contained more than 70% of NA values. Interestingly, this resulted in a dataset of 60 variables but with 100% complete cases. We also removed the dataset index values (X) after discovering that our decision tree model used them as the split variable. We used the variable importance (varImp) function throughout our testing to continue to identify variables that were, we felt, unduly influencing model results. In some cases, particularly with the random forest model runs, the varImp function returned timestamp values that had appended measurements, so we removed all timestamp values as well. We finally settled on a training set of 19622 observations using 55 variables. All of the variables that we removed from the training set were also removed from the test set.

```
str(wlTrain)
sum(complete.cases(wlTrain))

# Pare down variable list by checking for colinearity and removing the index va
lue
colinCheck <- nearZeroVar(wlTrain, names = TRUE)
colinCheck
# Actually do it
trimNZVTrain <- wlTrain[,!(colnames(wlTrain) %in% colinCheck)]
trimNZVTest <- wlTest[,!(colnames(wlTest) %in% colinCheck)]
# Remove other variables that are affecting the model
trimIdxTrain <- trimNZVTrain[,-1]
trimIdxTest <- trimNZVTest[,-1]
timeStamp <- grep("stamp", names(trimIdxTrain), value = TRUE)
trimStampTrain <- trimIdxTrain[,!(colnames(trimIdxTrain) %in% timeStamp)]
trimStampTest <- trimIdxTest[,!(colnames(trimIdxTest) %in% timeStamp)]

# subset to variables that are no more than 70% NA
excessNA <- names(trimStampTrain[, colSums(is.na(trimStampTrain)) < nrow(trimSt
ampTrain) * 0.7])
wlTrainFinal <- trimStampTrain[,colnames(trimStampTrain) %in% excessNA]
wlTestFinal <- trimStampTest[,colnames(trimStampTest) %in% excessNA]
```

Data Partitioning

We randomly divided our training set on the classe variable using 80% for training and reserving 20% for testing (this ultimately made the provided “test” set our validation set).

```
# very little data in test. Partition training set and use test set for validat
ion.
set.seed(12333)
inTrain <- createDataPartition(wlTrainFinal$classe, p = 0.8, list = FALSE)
training <- wlTrainFinal[inTrain,]
testing <- wlTrainFinal[-inTrain,]
```

Model Selection

Because this exercise was a 5-level classification problem, we focused our model selection on decision trees, random forests, and finally a gradient boosted model. For the decision tree and random forest models we ran three separate models each:

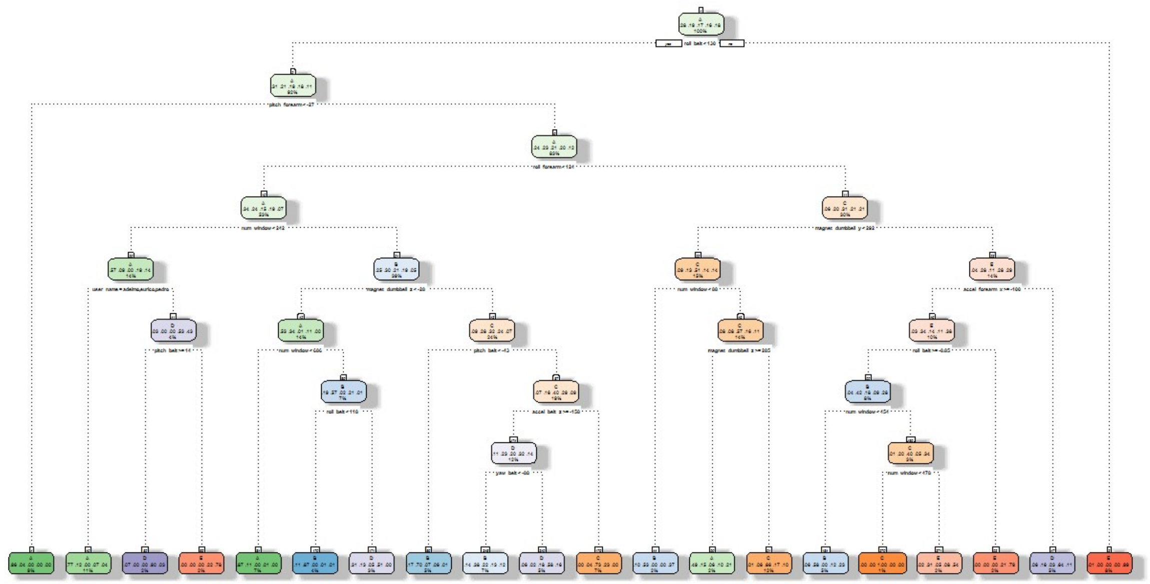
- Caret package method (no cross validation)
- Caret package method (5-fold cross validation)
- The standalone R packages (rpart and randomForest)

We only elected to run the Caret package’s “gbm” method, with no cross validation.

Results

Our first model was Caret’s rpart method which failed to produce any leafs representing Classes B and D as well as finishing with what would ultimately be the lowest accuracy (43%). Since we didn’t anticipate significant improvement, we next evaluated the standalone rPart package and the results were significantly better with an accuracy of 71.6% and all classes represented. However, this model assigned zero importance to many variables, including most of the gyroscope features. Despite this, we were pleased with the improvements and were comfortable with the depth of the overall tree.

```
# modFitRpart <- rpart(classe ~ ., data = training)
load(file = "MLmodFitRpart.rda")
fancyRpartPlot(modFitRpart)
```



Rattle 2016-Oct-08 15:07:32 kduff

The confusion matrix and predictions from this model are below. Note that predictions are for both the partitioned test set (3900 observations) and then the validation data (original 20 test observations).

```
predRPart <- predict(modFitRpart, testing, type = "class")
table(predRPart)
```

```
## predRPart
##    A    B    C    D    E 
## 1156  883  790  537  557
```

```
confusionMatrix(predRPart, testing$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   A   B   C   D   E
##          A 937 127   5  55  32
##          B 114 520  89  71  89
##          C   9  49 525 163  44
##          D  50  35  65 328  59
##          E   6  28   0  26 497
## 
## Overall Statistics
##                                           
##                Accuracy : 0.7155          
##                  95% CI : (0.7011, 0.7296)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.6394          
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8396   0.6851   0.7675  0.51011   0.6893
## Specificity            0.9220   0.8853   0.9182  0.93628   0.9813
## Pos Pred Value         0.8106   0.5889   0.6646  0.61080   0.8923
## Neg Pred Value         0.9353   0.9214   0.9492  0.90697   0.9335
## Prevalence             0.2845   0.1935   0.1744  0.16391   0.1838
## Detection Rate         0.2388   0.1326   0.1338  0.08361   0.1267
## Detection Prevalence   0.2947   0.2251   0.2014  0.13689   0.1420
## Balanced Accuracy      0.8808   0.7852   0.8429  0.72319   0.8353
```

```
# Predict on validation
predValRpart <- predict(modFitRpart, wlTestFinal, type = "class")
table(predValRpart)
```

```
## predValRpart
##  A  B  C  D  E 
## 11  3  3  2  1
```

We then began evaluating Random Forest models (which proved in the case of Caret to be computationally expensive and extremely time consuming). Both the Caret and standalone Random Forest models performed extremely well. In fact, their accuracies were identical at 99.6% and while their total misclassifications were identical (14 each), there were differences in which observations were misclassified (but both models erred the most with Classes C and D). Finally both models predicted the same results for our holdout validation set (the 20 original test observations).

Though we were satisfied with our Random Forest models, we continued our evaluation with Caret's "gbm" method, which returned only slightly less accuracy (98.9%). That model also returned the same results for the validation set.

```
## predValGBM
## A B C D E
## 7 8 1 1 3
```

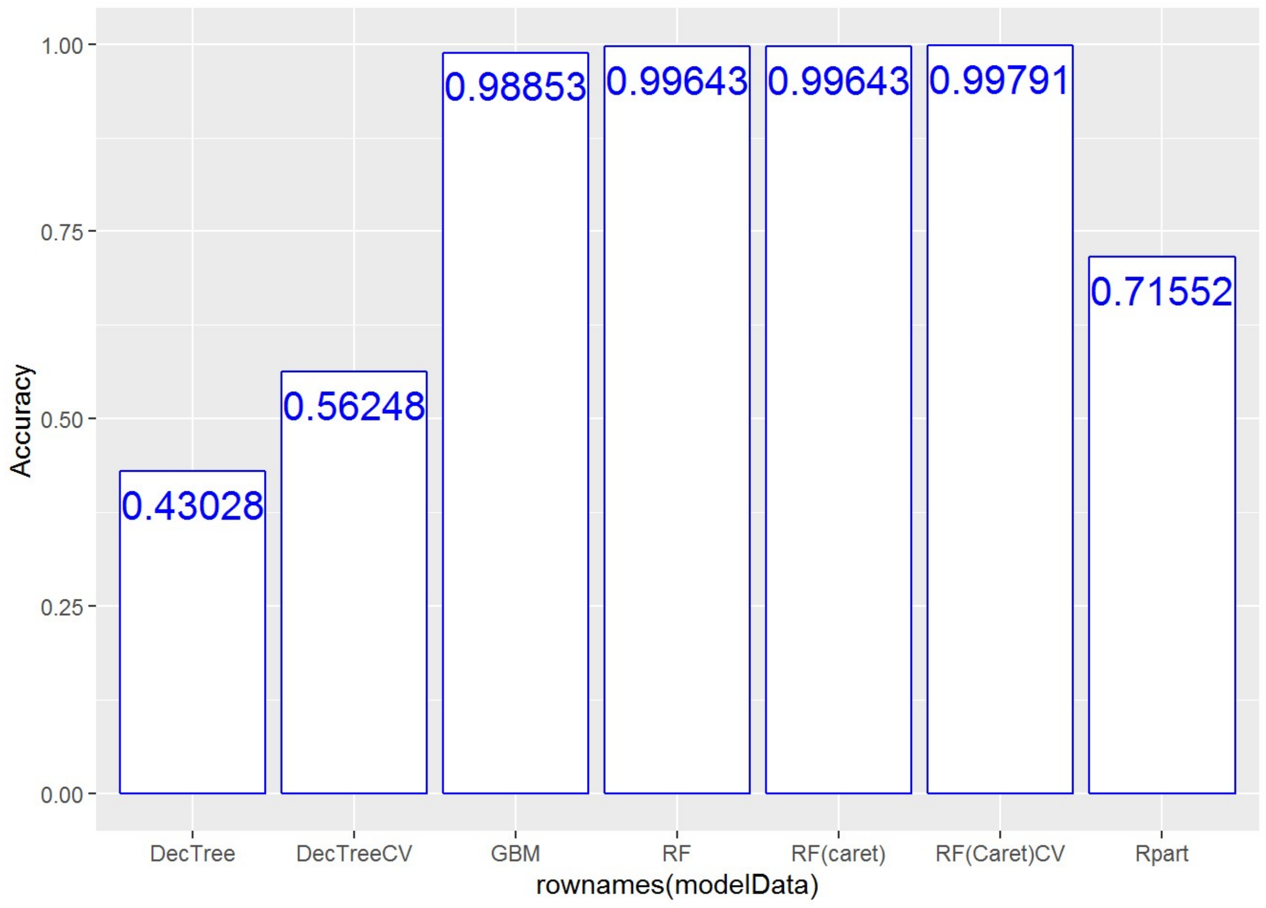
Our final tests were designed to add cross validation to the entire training set with the "rpart" and "rf" methods of the Caret package. Cross validation improved the decision tree accuracy results (56.2%), but failed to place any observations in three of the five classes (B, D and E) and we rejected it. The cross validation with Caret's "rf" method actually outperformed the other two Random Forest models, though only slightly with an overall accuracy of 99.8%. This model, too, returned the same predictions against the validation set as the Random Forest and Gradient Boosted Models.

The model fit for the cross validated Random Forest is presented below.

```
# Train without partitioning
# modFitRFCV <- train(classe ~ ., method = "rf", data = wlTrainFinal, trContro
l = trainControl(method = "cv", number = 5))
load(file = "MLmodFitRFCV.rda")
modFitRFCV
```

```
## Random Forest
##
## 19622 samples
##    54 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 15698, 15698, 15697, 15697, 15698
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9958719  0.9947781
##   30    0.9979104  0.9973569
##   58    0.9958209  0.9947136
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 30.
```

The overall results are graphically represented below:



Our final predictions for this model were:

```
# Predict on test
predTestRFCV <- predict(modFitRFCV, wlTestFinal)
predTestRFCV
```

```
##  [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
```

```
table(predTestRFCV)
```

```
## predTestRFCV
## A B C D E
## 7 8 1 1 3
```

Assumptions, Uncertainty, and Conclusion

We are comfortable with our selection of the cross-validated Random Forest model as the best performer, but unlike the models run against the training data's test partition, we were unable to explore the results in a confusion matrix. Still, given the consistent predictions on the validation sets between all three Random Forest models and the Gradient Boosted Model, we feel that if nothing else our models are consistent. Admittedly, we did little customization and wherever possible, relied on the default model methods. Our unfamiliarity with the dataset variables is also cause for concern. Our variable importance checks for all our models put the most importance on the "num_window" variable, which we suspect could be safely removed from the dataset. For this exercise we chose to let it remain.