Predicting Exercise Classification from Movement Tracker Data

Jerry Lakin

4/12/2021

Introduction

The wide proliferation of wearable fitness devices in recent years has made personal activity data accessible in large quantities. It is important to be able to detect not only the amount of activity performed by the wearer of a device but also the manner in which it was performed. In this project we will be using data from personal fitness devices to classify the method by which test participants perform barbell lifts.

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Loading Libraries and Data

First load the libraries and set our see.

```r
library(caret)
library(rattle)

set.seed(1010)
```

Next pull in the data.

```r
dat <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv', na.strings=c(""
testing_dat <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv')
```

Preprocessing Data & Setting up for Cross-Validation

Remove all null columns.

```r
dat <- dat[, colSums(is.na(dat)) == 0]
testing_dat <- testing_dat[, colSums(is.na(testing_dat)) == 0]
```

Next remove the first 7 columns containing metadata that will not be useful for classification.

```r
dat_f <- dat[, -c(1:7)]
testing <- testing_dat[, -c(1:7)]
```

Finally split the primary data set into training and validation sets. The validation set will be used to determine the quality of the models while the testing set will not be used until the final model has been selected.

```r
inTrain <- createDataPartition(y=dat_f$classe, p=0.7, list=FALSE)

training <- dat_f[inTrain,]
validation <- dat_f[-inTrain,]
```

Before we begin to train our models, we will want to set up our trainControl object which we will use to perform cross-validation.

```r
control <- trainControl(method = "cv", number=3, verboseIter=F)
```

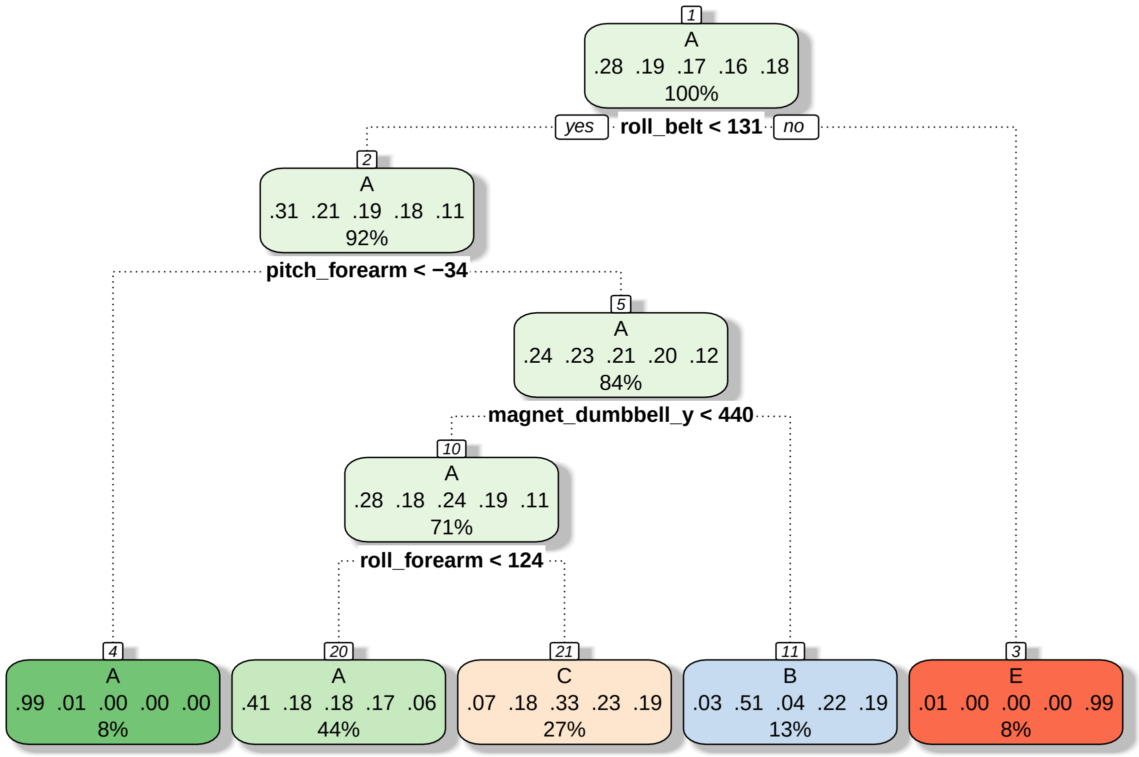
Decision Tree

As a first pass we will train and visualize a decision tree model and test it on our validation set. First we want to train the model with our training set.

```r
fit_trees <- train(classe ~ ., data=training, method="rpart", trControl=control, tuneLength=3)
```

Next we will take a look at the visualization to see how the decision tree makes predictions.

```r
fancyRpartPlot(fit_trees$finalModel)
```



Rattle 2021−Apr−13 00:01:15 gerar

In the figure we can see the variables and thresholds that the model has chosen in order to make decisions. For example if the roll_belt variable is greater than 131, the trial will be classified as E. if the roll_belt is less than 131 and the pitch_forearm variable is less than -34, the trial will be classified as A, etc.

Next we will use the model to make predictions on our validation set and see how it performs by looking at the confusion matrix.

```
pred_trees <- predict(fit_trees, validation)
cm_trees <- confusionMatrix(pred_trees, factor(validation$classe))
cm_trees
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1505  488  485  444  147
##          B   22  374   33  177  146
##          C  144  277  508  343  286
##          D    0    0    0    0    0
##          E    3    0    0    0  503
##
## Overall Statistics
##
##                Accuracy : 0.4911
##                  95% CI : (0.4782, 0.5039)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3347
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8990  0.32836  0.49513   0.0000  0.46488
## Specificity            0.6286  0.92035  0.78391   1.0000  0.99938
## Pos Pred Value         0.4904  0.49734  0.32606      NaN  0.99407
## Neg Pred Value         0.9400  0.85096  0.88029   0.8362  0.89236
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2557  0.06355  0.08632   0.0000  0.08547
## Detection Prevalence   0.5215  0.12778  0.26474   0.0000  0.08598
## Balanced Accuracy      0.7638  0.62436  0.63952   0.5000  0.73213
```

Random Forest

Next we will train a model using a random forest. This is a more robust model and should therefore provide a more accurate prediction.

```
fit_rf <- train(classe ~ ., data=training, method="rf", trControl=control, tuneLength=3)
```

Next we use the model to generate predictions on our validation set and look at the results.

```
pred_rf <- predict(fit_rf, validation)
cm_rf <- confusionMatrix(pred_rf, factor(validation$classe))
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    2    0    0    0
##          B    2 1134   11    0    0
##          C    0    3 1013   11    0
##          D    0    0    2  953    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9925, 0.9964)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9933
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9956   0.9873   0.9886   1.0000
## Specificity            0.9995   0.9973   0.9971   0.9996   1.0000
## Pos Pred Value         0.9988   0.9887   0.9864   0.9979   1.0000
## Neg Pred Value         0.9995   0.9989   0.9973   0.9978   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1927   0.1721   0.1619   0.1839
## Detection Prevalence   0.2845   0.1949   0.1745   0.1623   0.1839
## Balanced Accuracy      0.9992   0.9964   0.9922   0.9941   1.0000
```

We can see that this model is highly accurate compared to our decision tree model, with accuracy exceeding 99%. This should be a much better model for use to generate final predictions on our testing set.

Final Results

Finally, we use our random forest model to generate predicions on the testing set, containing data that neither of our models has seen to this point.

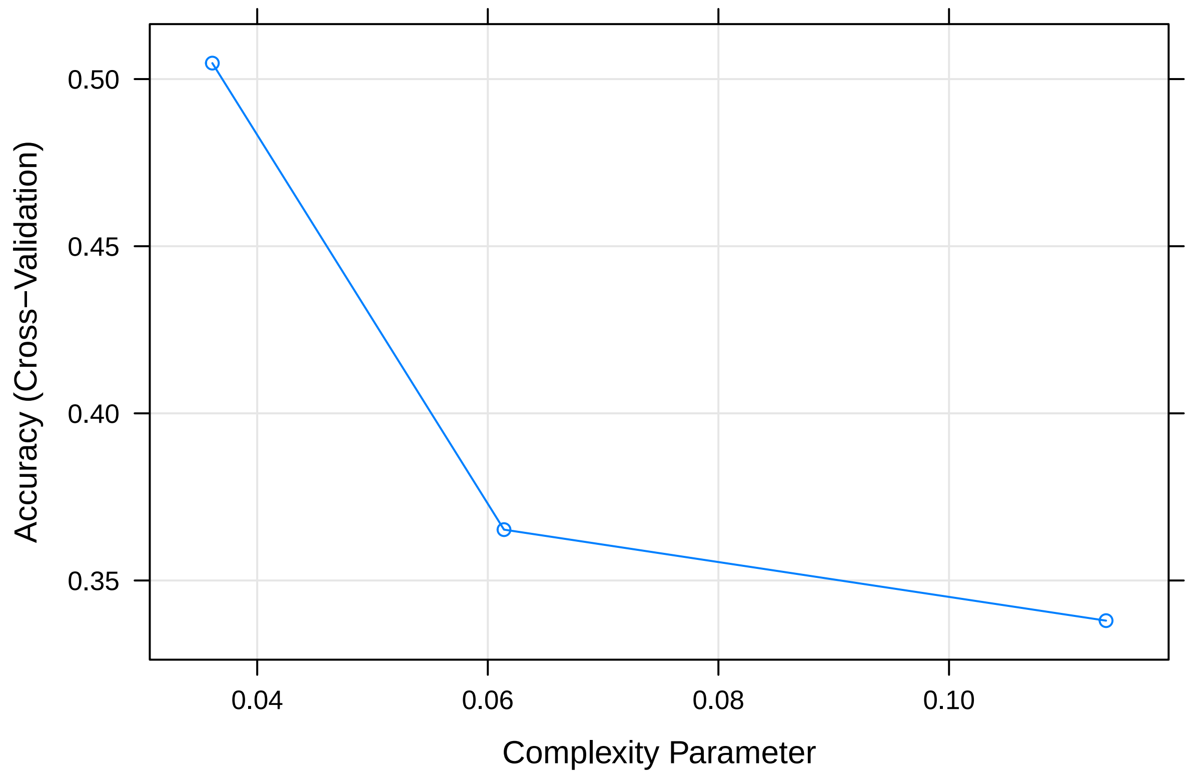
```
pred_final <- predict(fit_rf, testing)
pred_final
```

```
##  [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
```

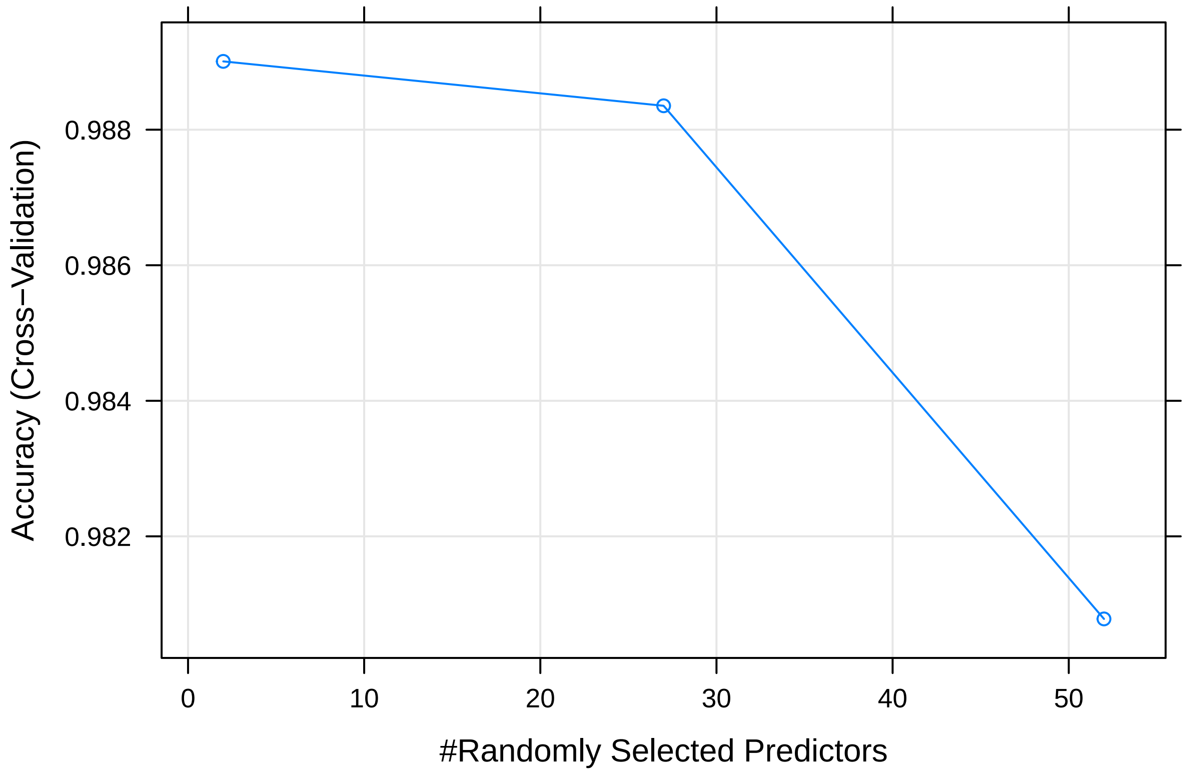
Appendix

For cross-validation purposes we will plot our models' accuracy.

```r
plot(fit_trees)
```



```r
plot(fit_rf)
```


Accuracy (Cross−Validation)
0.988
0.986
0.984
0.982
0
10
20
30
40
50
#Randomly Selected Predictors