Machine Learning Project ASmyth

Abigail Smyth

June 1, 2020

Machine Learning Course Project

We will be using data collected from devices such as Jawbone Up, Nike FuelBand, and Fitbit to build a model to predict the manner in which an exercise was done. We will be reviewing data collected on the belt, forearm, arm, and dumbell of 6 participants that were asked to perform barbell lifts correctly and incorrectly in 5 different ways. We will use this data to build a prediction model to determine the classe variable.

Summary

We begin with importing the training data set, reviewing the data, removing columns that contain N/A to remove insignificant data. This brings the dataset from 160 columns down to 60 columns. In reviewing the research paper (http://groupware.les.inf.puc-rio.br/har#literature), it is clear that the first 7 columns have no impact on predicting the class and we will remove them as well, bringing our dataset down to 52 predictors. We then split the data set into a training and test set (.70 / .30).

Looking at the classe variable, we see that the highest percentage - 28% of the exercises - fall in classe A (or the classe where the exercise was correctly completed).

Models compared were Decision Tree, Random Forest and Naive Bayes. Random forest had the highest accuracy at 99%.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.3
```

```
library(rattle)
```

```
## Warning: package 'tibble' was built under R version 3.5.3
```

```
train <- read.csv("pml-training.csv", na.strings=c("NA", "#DIV/0!", "", " "));
test  <- read.csv("pml-testing.csv", na.strings=c("NA", "#DIV/0!", "", " "));


train <- train[-c(1:7)];          # remove first 7 columns
test  <- test[-c(1:7,160)];       # remove first 7 and result column

# function to verify existence of atleast one NA value in a given column
is.na.column = function(col){
   any(is.na(col))
}

cols <- names(train);
non.empty.columns <- which(!sapply(train,is.na.column));
train <- train[,non.empty.columns]

cols <- names(test);
non.empty.columns <- which(!sapply(test,is.na.column));
```

```
test <- test[,non.empty.columns]

dim(train);
```

```
## [1] 19622    53
```

```
summary(train$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
dim(test);
```

```
## [1] 20 52
```

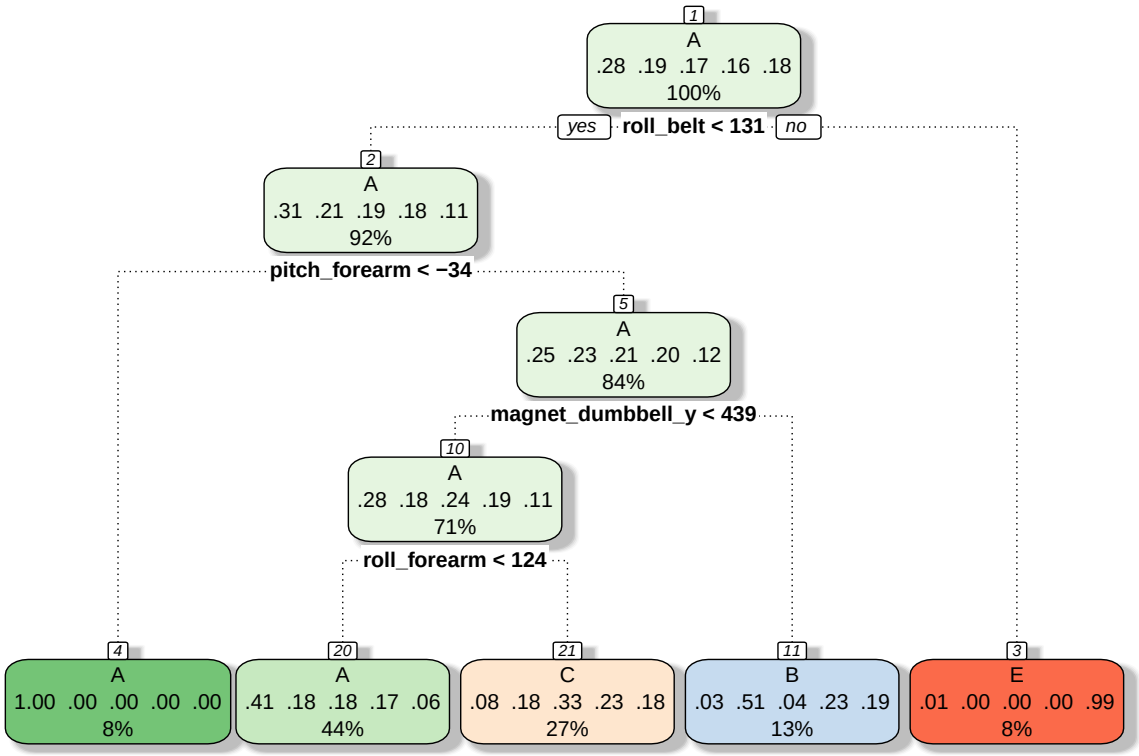
```
trainIndex <- createDataPartition(y=train$classe, p=0.70, list=FALSE);
trainSet <- train[trainIndex,];
testSet <- train[-trainIndex,];
```

Decision Tree Model - ~49% accuracy

```
modTree <- train(classe ~ ., data=trainSet, method="rpart")
print(modTree$finalModel)
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 12573 8677 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -34.35 1080    1 A (1 0.00093 0 0 0) *
##      5) pitch_forearm>=-34.35 11493 8676 A (0.25 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 438.5 9723 6957 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 123.5 6074 3593 A (0.41 0.18 0.18 0.17 0.06) *
##         21) roll_forearm>=123.5 3649 2444 C (0.078 0.18 0.33 0.23 0.18) *
##       11) magnet_dumbbell_y>=438.5 1770  874 B (0.029 0.51 0.042 0.23 0.19) *
##    3) roll_belt>=130.5 1164   10 E (0.0086 0 0 0 0.99) *
```

```
fancyRpartPlot(modTree$finalModel)
```



Rattle 2020−Jun−03 13:02:20 Abigail

```
predTree <- predict(modTree, testSet)
cmTree <- confusionMatrix(predTree, testSet$classe)
print(cmTree);
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1513  471  469  432  161
##          B   36  397   35  162  151
##          C  121  271  522  370  293
##          D    0    0    0    0    0
##          E    4    0    0    0  477
##
## Overall Statistics
##
##                Accuracy : 0.4943
##                  95% CI : (0.4815, 0.5072)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3393
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9038  0.34855   0.5088   0.0000  0.44085
## Specificity            0.6360  0.91909   0.7829   1.0000  0.99917
## Pos Pred Value         0.4967  0.50832   0.3310      NaN  0.99168
## Neg Pred Value         0.9433  0.85462   0.8830   0.8362  0.88805
## Prevalence             0.2845  0.19354   0.1743   0.1638  0.18386
## Detection Rate         0.2571  0.06746   0.0887   0.0000  0.08105
## Detection Prevalence   0.5176  0.13271   0.2680   0.0000  0.08173
## Balanced Accuracy      0.7699  0.63382   0.6458   0.5000  0.72001
```

Random Forest Model - ~99% accuracy

```
modRF <- randomForest(classe ~. , data=trainSet, method="class")
predRF <- predict(modRF, testSet)
cmRF <- confusionMatrix(predRF, testSet$classe)
print(cmRF);
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    3    0    0    0
##          B    0 1135    3    0    0
##          C    0    1 1019    5    0
##          D    0    0    4  959    4
##          E    0    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.9948, 0.9979)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9957
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   0.9932   0.9948   0.9963
## Specificity            0.9993   0.9994   0.9988   0.9984   1.0000
## Pos Pred Value         0.9982   0.9974   0.9941   0.9917   1.0000
## Neg Pred Value         1.0000   0.9992   0.9986   0.9990   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1732   0.1630   0.1832
## Detection Prevalence   0.2850   0.1934   0.1742   0.1643   0.1832
## Balanced Accuracy      0.9996   0.9979   0.9960   0.9966   0.9982
```

Naive Bayes Model - ~75% accuracy

```
#modNB <- train(classe ~., data=trainSet, method="nb")
#predNB <- predict(modNB, testSet)
#cmNB <- confusionMatrix(predNB, testSet$classe)
#print(cmNB)
```

Naive Bayes Confusion Matrix

Note - My laptop only has 4Gb RAM and this takes hours to run. I had to rerun and didn't feel like taking hours to generate the HTML - so copied & pasted the results below from the console.

Confusion Matrix and Statistics

```
      Reference
```

Prediction A B C D E A 1240 88 53 57 28 B 54 802 77 4 96 C 151 161 846 188 64 D 200 72 42 670 36 E 29 16 8 45 858

Overall Statistics

```
           Accuracy : 0.7504
             95% CI : (0.7391, 0.7614)
No Information Rate : 0.2845
P-Value [Acc > NIR] : < 2.2e-16

              Kappa : 0.6861
```

Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

```
                     Class: A Class: B Class: C Class: D Class: E
```

Sensitivity 0.7407 0.7041 0.8246 0.6950 0.7930 Specificity 0.9463 0.9513 0.8839 0.9289 0.9796 Pos Pred Value 0.8458 0.7764 0.6000 0.6569 0.8975 Neg Pred Value 0.9018 0.9305 0.9598 0.9396 0.9546 Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839 Detection Rate 0.2107 0.1363 0.1438 0.1138 0.1458 Detection Prevalence 0.2491 0.1755 0.2396 0.1733 0.1624 Balanced Accuracy 0.8435 0.8277 0.8542 0.8119 0.8863

Predicting the Test Results

```
predTest <- predict(modRF, test)
predTest
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```