FitFam

Offiong Eyo

July 23, 2017

Executive Summary

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

The purpose of this document is to predict the manner in which 6 participants performed their exercises. This is the "classe" variable in the training set. The machine learning algorithm described here is applied to the 20 test cases available in the test data.

Data Loading & Pre-Processing

Here the data is loaded and pre-processed. My approach reserves the test sample for use as a validation set, and partitions the training data 70/30 for use as both the training set and cross-validated test set.

```
library(rpart)
library(caret)
library(randomForest)

training <- read.csv('C:\\Users\\AKKAN\\Downloads\\pml-training.csv')
testing  <- read.csv('C:\\Users\\AKKAN\\Downloads\\pml-testing.csv')
set.seed(12345)   # seed is set for reproducibility of results

#create training and testing partitions from the training dataset
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]
dim(TrainSet)

## [1] 13737   160
```

These 160 variables are further reduced after accounting for Near Zero Variance (NZV) variables, identity variables and majority NA variables. Keeping in mind that all transformations performed on training data must also be performed on cross-validation testing data.

```
NZV <- nearZeroVar(TrainSet)  #identifying variables with near zero variance
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]
dim(TrainSet)

## [1] 13737   106

MajNA    <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95  #identifying majority NA variables
TrainSet <- TrainSet[, MajNA==F]
TestSet  <- TestSet[, MajNA==F]
dim(TrainSet)

## [1] 13737    59

TrainSet <- TrainSet[, -(1:5)]    #identifying identity variables
TestSet  <- TestSet[, -(1:5)]
dim(TrainSet)

## [1] 13737    54
```

Choosing the Prediction Model

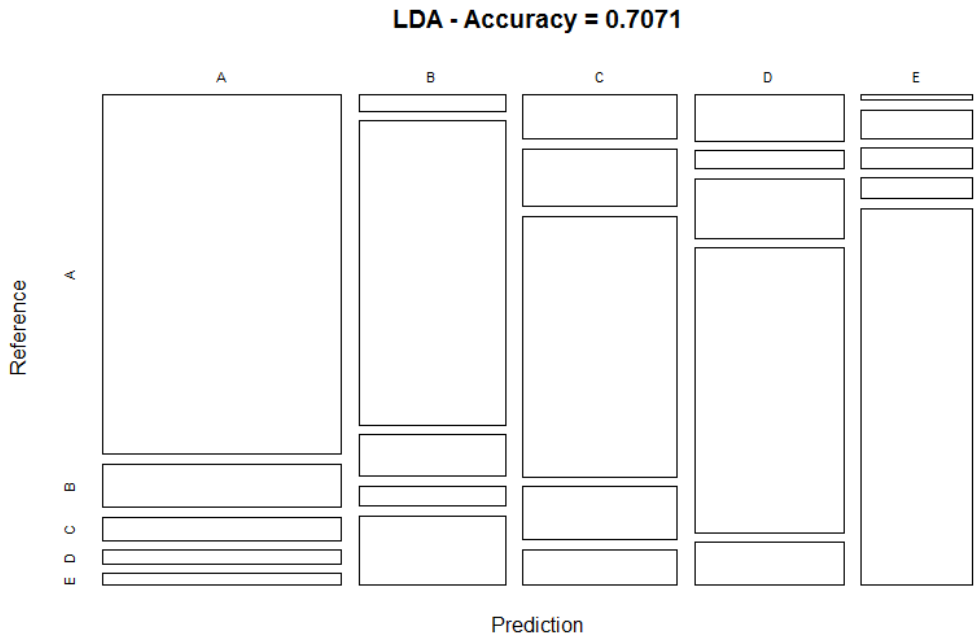
The candidate methods for building the prediction model are Linear Discriminant Analysis, Decision Tree, and Random Forests Methods. After a comprehensive comparison of their confusion matrices, one will be selected on the strength of their reported accuracies.

Linear Discriminant Analysis

```
lineFit <- train(classe ~ ., data='TrainSet, method='lda')

## Loading required package: MASS

linePredict <- predict(lineFit, newdata=TestSet)  #prediction on test dataset
lineMatrix <- confusionMatrix(linePredict, TestSet$classe)
print(lineMatrix)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1404  168   92   52   44
##          B   39  728   99   47  166
##          C  110  146  655  133   89
##          D  112   45  144  694  103
##          E    9   52   36   38  680
##
## Overall Statistics
##
##                Accuracy : 0.7071
##                  95% CI : (0.6952, 0.7187)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6291
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity          0.8387   0.6392   0.6384   0.7199   0.6285
## Specificity          0.9155   0.9260   0.9016   0.9179   0.9719
## Pos Pred Value       0.7977   0.6747   0.5781   0.6321   0.8344
## Neg Pred Value       0.9345   0.9145   0.9219   0.9436   0.9207
## Prevalence           0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate       0.2386   0.1237   0.1113   0.1179   0.1155
## Detection Prevalence 0.2991   0.1833   0.1925   0.1866   0.1385
## Balanced Accuracy    0.8771   0.7826   0.7700   0.8189   0.8002
```

```
plot(lineMatrix$table, col = lineMatrix$byClass, main = paste("LDA - Accuracy =", round(lineMatrix$overall['Accuracy'], 4)))
```



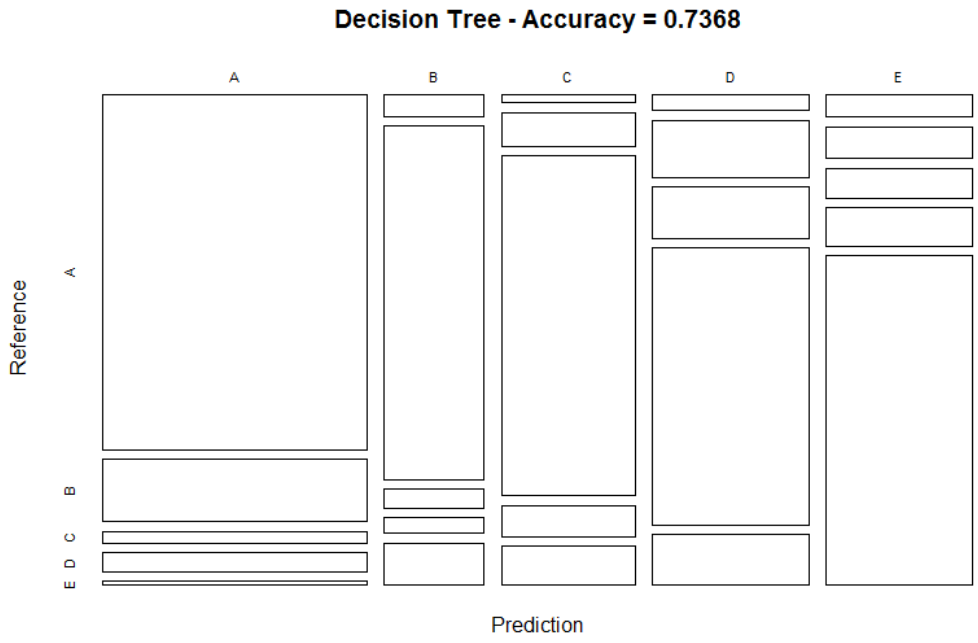
Decision Trees

```
treeFit <- rpart(classe ~ ., data=TrainSet, method='class')
treePredict <- predict(treeFit, newdata=TestSet, type='class')  #prediction on test dataset
treeMatrix <- confusionMatrix(treePredict, TestSet$classe)
print(treeMatrix)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    A    B    C    D    E
##          A 1530  269   51   79   16
##          B   35  575   31   25   68
##          C   17   73  743   68   84
##          D   39  146  130  702  128
##          E   53   76   71   90  786
##
## Overall Statistics
##
##               Accuracy : 0.7368
##                 95% CI : (0.7253, 0.748)
##    No Information Rate : 0.2845
##    P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.6656
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140  0.50483   0.7242   0.7282   0.7264
## Specificity            0.9014  0.96650   0.9502   0.9100   0.9396
## Pos Pred Value         0.7866  0.78338   0.7543   0.6131   0.7305
## Neg Pred Value         0.9635  0.89051   0.9422   0.9447   0.9384
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2600  0.09771   0.1263   0.1193   0.1336
## Detection Prevalence   0.3305  0.12472   0.1674   0.1946   0.1828
## Balanced Accuracy      0.9077  0.73566   0.8372   0.8191   0.8330
```

```
plot(treeMatrix$table, col = treeMatrix$byClass, main = paste("Decision Tree - Accuracy =",
round(treeMatrix$overall['Accuracy'], 4)))
```



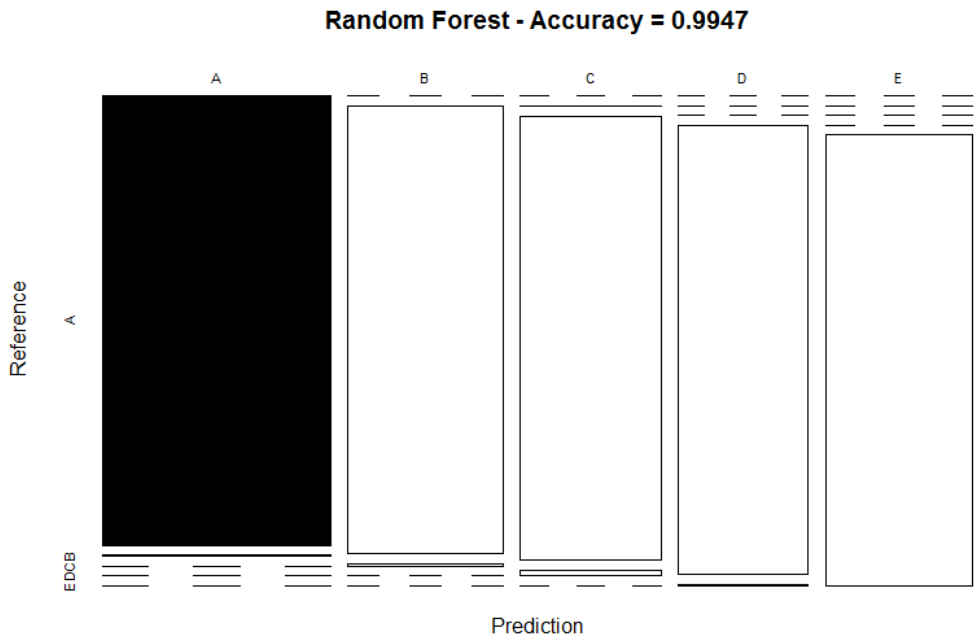
Random Forests

```
forestFit <- randomForest(classe ~ ., data=TrainSet)
forestPredict <- predict(forestFit, newdata=TestSet)  #prediction on test dataset
forestMatrix <- confusionMatrix(forestPredict, TestSet$classe)
print(forestMatrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    6    0    0    0
##          B    0 1132    7    0    0
##          C    0    1 1019   13    0
##          D    0    0    0  951    4
##          E    0    0    0    0 1078
##
## 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            1.0000   0.9939   0.9932   0.9865   0.9963
## Specificity            0.9986   0.9985   0.9971   0.9992   1.0000
## Pos Pred Value         0.9964   0.9939   0.9864   0.9958   1.0000
## Neg Pred Value         1.0000   0.9985   0.9986   0.9974   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1924   0.1732   0.1616   0.1832
## Detection Prevalence   0.2855   0.1935   0.1755   0.1623   0.1832
## Balanced Accuracy      0.9993   0.9962   0.9951   0.9929   0.9982
```

```
plot(forestMatrix$table, col = forestMatrix$byClass, main = paste("Random Forest - Accuracy =",
round(forestMatrix$overall['Accuracy'], 4)))
```



With the highest accuracy of 99%, the Random Forests prediction is selected to predict with the validation (original test) dataset as shown:

```
testPredict <- predict(forestFit, newdata=testing)  #prediction on test dataset
print(testPredict)
```

```
##  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
```

This ends the report.