Course Project

Veronica Vaca

05 de Abril de 2018

Introduction

The goal of this project is to predict the manner in which some people perform barbell lifts. We will use some variables in the data to do it. We are going to show:

- How the model was built.
- How was used cross validation in it.
- Expected out of sample error.
- Conclusion and explanation.
- Predict 20 different test cases.

Data

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

The data for this project come from this source:
http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

Now we are going to download the data

```
#Download the data
if(!file.exists("pml-
training.csv")){download.file("https://d396qusza40orc.cloudfront.net/predmach
learn/pml-training.csv", destfile = "pml-training.csv")}

if(!file.exists("pml-
testing.csv")){download.file("https://d396qusza40orc.cloudfront.net/predmachl
earn/pml-testing.csv", destfile = "pml-testing.csv")}


#Read the training data and replace empty values by NA
trainingDataSet<- read.csv("pml-training.csv", sep=",", header=TRUE,
na.strings = c("NA","",'#DIV/0!'))
testingDataSet<- read.csv("pml-testing.csv", sep=",", header=TRUE, na.strings
= c("NA","",'#DIV/0!'))
```

Cleaning the data

Removing variables that are not usable that have nearly zero variance, variables that are almost always NA, and variables that don't make intuitive sense for prediction.

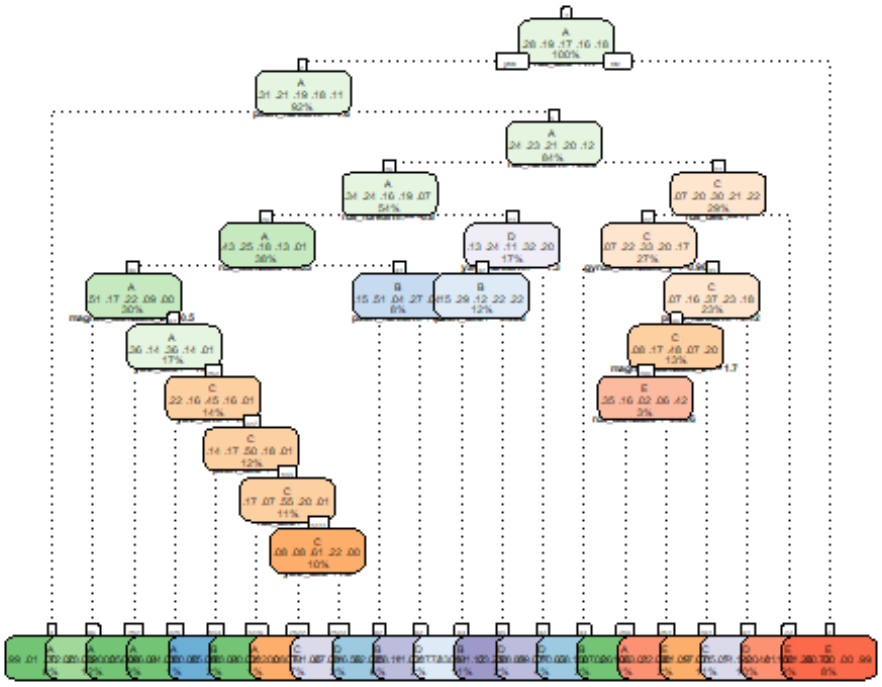
Building the model

For reproducibility we are going to set a seed and then subseting for having a training and a test set.

```
set.seed(2000)

idxTrain<- createDataPartition(pre_trainingDataSet$classe, p=3/4, list=FALSE)
training<- pre_trainingDataSet[idxTrain, ]
validation <- pre_trainingDataSet[-idxTrain, ]
dim(training) ; dim(validation)

## [1] 14718    28

## [1] 4904   28
```

We will start with the Decision Trees

```
modFitdt<-rpart(classe ~ ., data=training, method="class")
fancyRpartPlot(modFitdt,cex=.3,under.cex=0.2,shadow.offset=0)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



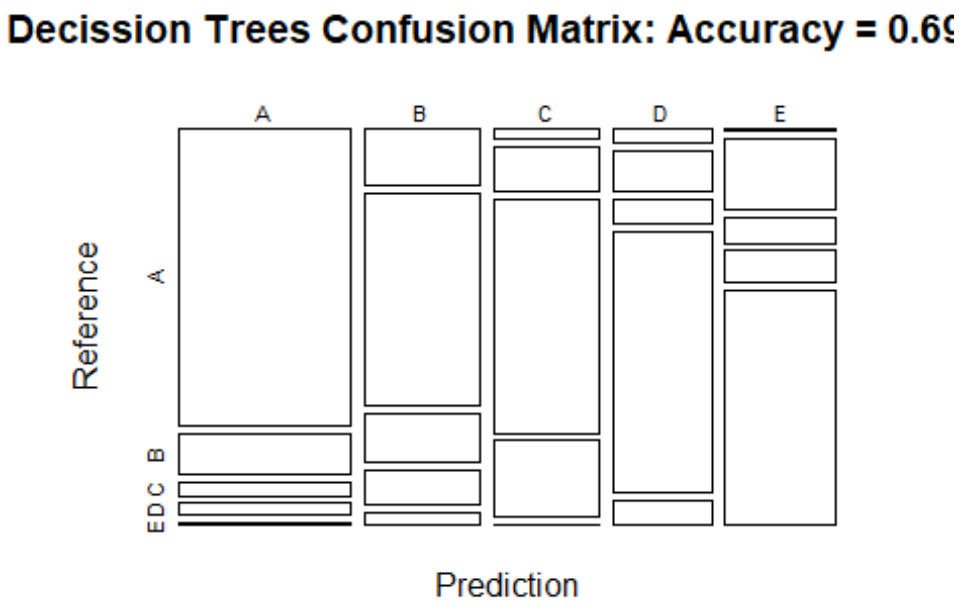
Now we'll see the accuracy

```
predictiondt <- predict(modFitdt, validation, type = "class")

cmtr<-confusionMatrix(validation$classe,predictiondt)
cmtr
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1139  159   48   46    3
##          B  149  552  130   89   29
##          C   25  107  546  176    1
##          D   32   91   55  574   52
##          E    8  174   63   77  579
##
## Overall Statistics
##
##                Accuracy : 0.6913
##                  95% CI : (0.6781, 0.7042)
##     No Information Rate : 0.2759
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6101
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8418   0.5097   0.6485   0.5967   0.8720
## Specificity            0.9279   0.8961   0.9239   0.9417   0.9241
## Pos Pred Value         0.8165   0.5817   0.6386   0.7139   0.6426
## Neg Pred Value         0.9390   0.8657   0.9269   0.9054   0.9788
## Prevalence             0.2759   0.2208   0.1717   0.1962   0.1354
## Detection Rate         0.2323   0.1126   0.1113   0.1170   0.1181
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.8849   0.7029   0.7862   0.7692   0.8980
```

We have an accuracy of 0.69

```
plot(cmtr$table, col = cmtr$byClass, main = paste("Decission Trees Confusion
Matrix: Accuracy =", round(cmtr$overall['Accuracy'], 4)))
```



Random Forest

Then we'll test the Random Forest for comparisson

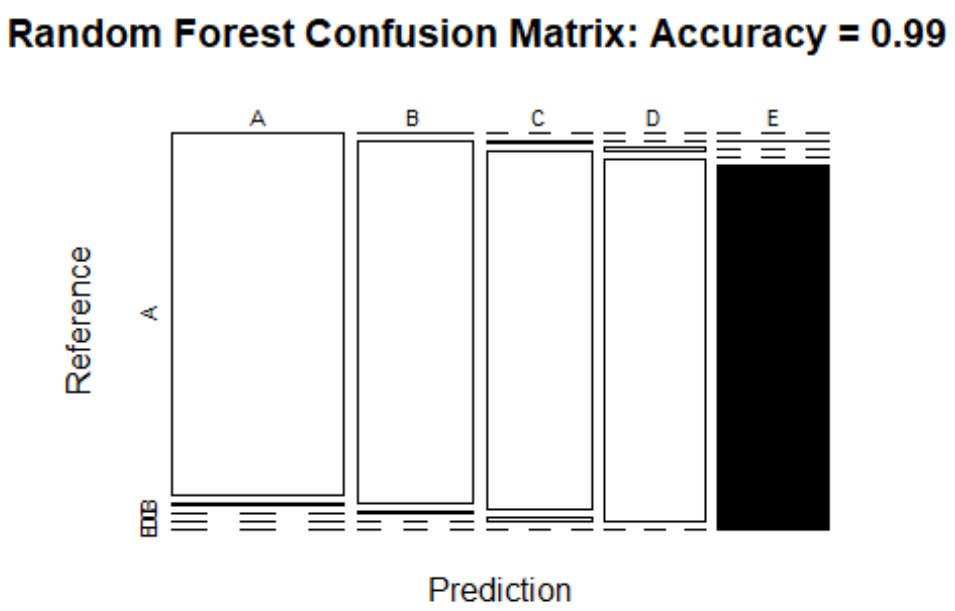
```
set.seed(4444)
modFitrf <- randomForest(classe ~ ., data=training)
predictionrf <- predict(modFitrf, validation, type = "class")
cmrf2 <- confusionMatrix(predictionrf, validation$classe)
cmrf2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    7    0    0    0
##          B    2  934    3    0    0
##          C    0    6  845    7    0
##          D    0    0    7  797    0
##          E    0    2    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9931
##                  95% CI : (0.9903, 0.9952)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.9912
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9842   0.9883   0.9913   1.0000
## Specificity            0.9980   0.9987   0.9968   0.9983   0.9995
## Pos Pred Value         0.9950   0.9947   0.9848   0.9913   0.9978
## Neg Pred Value         0.9994   0.9962   0.9975   0.9983   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1905   0.1723   0.1625   0.1837
## Detection Prevalence   0.2855   0.1915   0.1750   0.1639   0.1841
## Balanced Accuracy      0.9983   0.9915   0.9925   0.9948   0.9998
```

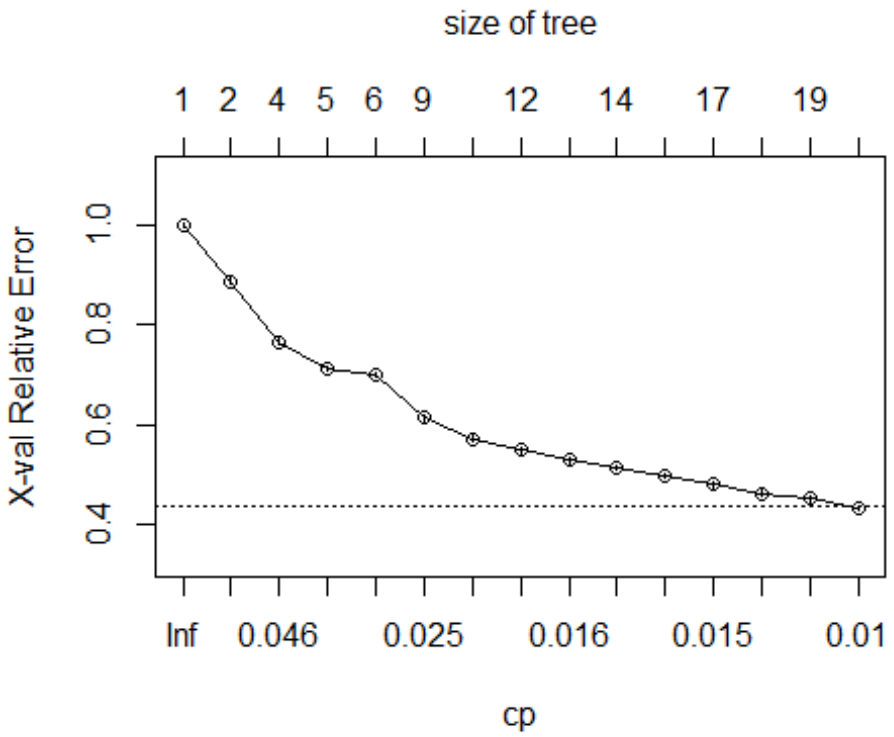
The random forest model has a 99.4% accuracy, far superior to the rpart method. The specificity and sensitivity is in the high 90s for all variables.

```
plot(cmrf2$table, col = cmrf2$byClass, main = paste("Random Forest Confusion
Matrix: Accuracy =", round(cmrf2$overall['Accuracy'], 4)))
```



Cross Validation

The cross-validated error for the decision tree is scaled down for easier reading; the error bars on the plot show one standard deviation of the x-validated error.



For random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally , during the run.

In Sample & Out of Sample Error

The in sample error is error rate when the model is used to predict the training set it is based off. This error is going to be much less than the model predicting another dataset (out of sample error).

```
insampledt <- predict(modFitdt, training, type = "class")
confusionMatrix(insampledt, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3411  460   49   80   14
##          B  467 1648  337  344  467
##          C  135  347 1667  169  217
##          D  152  328  514 1687  330
##          E   20   65    0  132 1678
##
## Overall Statistics
##
##                Accuracy : 0.6856
##                  95% CI : (0.6781, 0.6931)
##     No Information Rate : 0.2843
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6032
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8151   0.5787   0.6494   0.6994   0.6201
## Specificity            0.9428   0.8639   0.9286   0.8924   0.9819
## Pos Pred Value         0.8498   0.5051   0.6576   0.5603   0.8855
## Neg Pred Value         0.9277   0.8952   0.9261   0.9381   0.9198
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2318   0.1120   0.1133   0.1146   0.1140
## Detection Prevalence   0.2727   0.2217   0.1722   0.2046   0.1288
## Balanced Accuracy      0.8789   0.7213   0.7890   0.7959   0.8010
```

```
insamplerf <- predict(modFitrf, training, type = "class")
confusionMatrix(insamplerf, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
```

```
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

We have for decission trees that the accuracy is over 60% the in sample error is by 40% when we use random forests the submitted answer resulted in 100% with 0% in sample error. This could be a sign of overfitting.

The out of sample error will be tested with other samples of subjets.

Conclusion

Random Forest was a superior model for prediction of exercise quality compared to rpart decission trees, had over 99% accuracy and fitted well to other subsamples of the data. However, the algorithm may not have as high of accuracy on other samples, particularly ones with different subjects.