My Prediction Assignment Writeup

Patricio Cofre

October 22, 2016

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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Libraries used

```
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
library(e1071)
```

Loading data sets

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

The training set consists of 19622 observations of 160 variables The testing set consists of 20 observations of 160 variables

Cleaning data

Columns in the orignal training and testing datasets that are mostly filled with missing values are then removed count the number of missing values in each column of the full training dataset

```
training <-training[,colSums(is.na(training)) == 0]
testing <-testing[,colSums(is.na(testing)) == 0]
```

```
training   <-training[,-c(1:7)]
testing <-testing[,-c(1:7)]

dim ( training )
```

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

```
dim ( testing  )
```

```
## [1] 20 53
```

Diagnoses predictors that have one unique value or predictors that are have both of the following characteristics

```
ColumnsZVar <- nearZeroVar(training, saveMetrics = TRUE)
training <- training[, ColumnsZVar$nzv==FALSE]
training$classe = factor(training$classe)
```

Partitioning the training data This validation dataset will allow us to perform cross validation when developing our model.

Partitioning the training data set to allow cross-validation

```
set.seed(1234)
subTrain <- createDataPartition(y=training$classe, p=.75, list=FALSE)

TheTraining <- training[subTrain, ]
TheTesting <- training[-subTrain, ]
```

Dataset contains 59 variables, and the last column containing the 'class' variable we are trying to predict.

Modelprediction 1 : Using Decision Tree

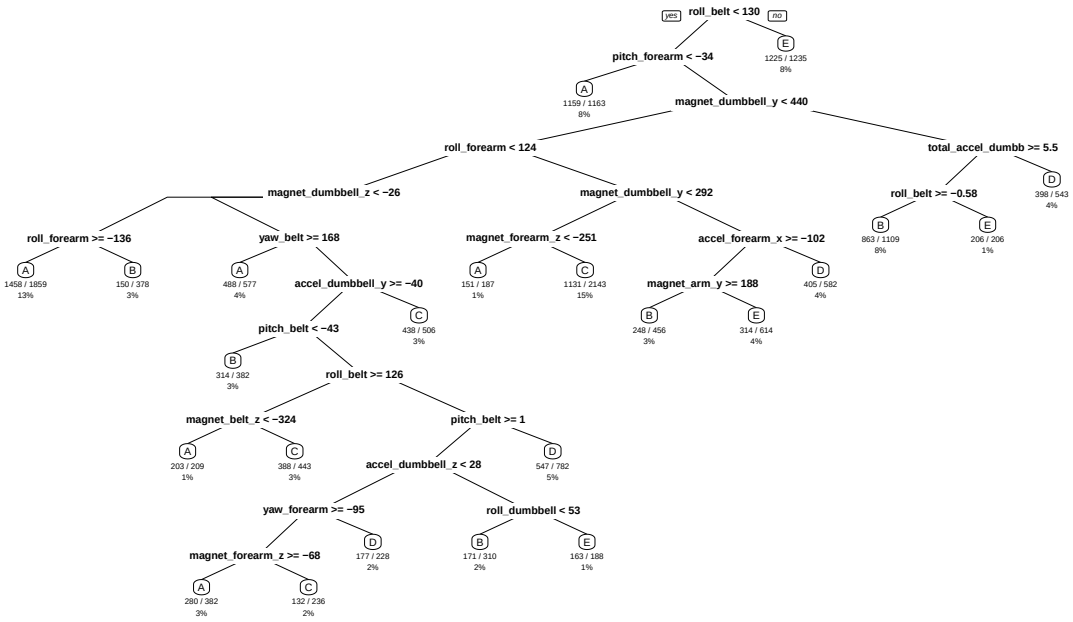
```
modelDT <- rpart(classe ~ ., data=TheTraining, method="class")
```

```
predictionDT <- predict(modelDT, TheTesting, type ="class")
```

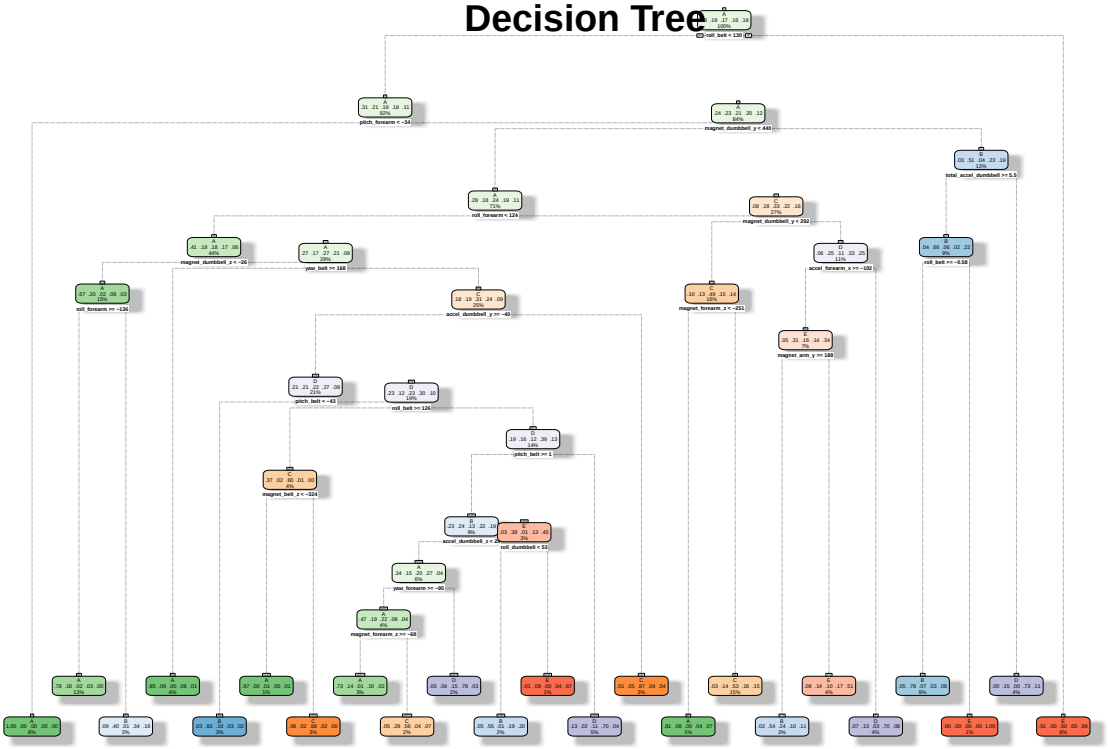
Plot of the Decision Tree

```
rpart.plot(modelDT, main="Decision Tree ", extra=102, under=TRUE, faclen=0)
```

Decision Tree




```
fancyRpartPlot (modelDT, main="Decision Tree")
```



Test results on our subTesting dataset

```
confusionMatrix(predictionDT,TheTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1235  157   16   50   20
##          B   55  568   73   80  102
##          C   44  125  690  118  116
##          D   41   64   50  508   38
##          E   20   35   26   48  625
##
## Overall Statistics
##
##                Accuracy : 0.7394
##                  95% CI : (0.7269, 0.7516)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6697
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8853   0.5985   0.8070   0.6318   0.6937
## Specificity            0.9307   0.9216   0.9005   0.9529   0.9678
## Pos Pred Value         0.8356   0.6469   0.6313   0.7247   0.8289
## Neg Pred Value         0.9533   0.9054   0.9567   0.9296   0.9335
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2518   0.1158   0.1407   0.1036   0.1274
## Detection Prevalence   0.3014   0.1790   0.2229   0.1429   0.1538
## Balanced Accuracy      0.9080   0.7601   0.8537   0.7924   0.8307
```

The Confusion Matrix achieved 0.7394 % accuracy.

Modelprediction 2 : Using Random Forest

```
modelRF <- randomForest(classe ~. , data=TheTraining, method="class")
print (modelRF)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = TheTraining, method = "class")
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.43%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4182    2    0    0    1 0.0007168459
## B   14 2830    4    0    0 0.0063202247
## C    0   11 2553    3    0 0.0054538372
## D    0    0   18 2392    2 0.0082918740
## E    0    1    3    5 2697 0.0033259424
```

Predicting:

```
predictionRF <- predict(modelRF, TheTesting, type = "class")
```

Test results on subTesting dataset

```
confusionMatrix(predictionRF, TheTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1394    3    0    0    0
##          B    1  944   10    0    0
##          C    0    2  843    6    0
##          D    0    0    2  798    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9951
##                  95% CI : (0.9927, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9938
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9947   0.9860   0.9925   1.0000
## Specificity            0.9991   0.9972   0.9980   0.9995   1.0000
## Pos Pred Value         0.9979   0.9885   0.9906   0.9975   1.0000
## Neg Pred Value         0.9997   0.9987   0.9970   0.9985   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1925   0.1719   0.1627   0.1837
## Detection Prevalence   0.2849   0.1947   0.1735   0.1631   0.1837
## Balanced Accuracy      0.9992   0.9960   0.9920   0.9960   1.0000
```

The Confusion Matrix achieved 99.51% accuracy.

Decision & Conclusion

Random Forest algorithm performed better than Decision Trees. Accuracy for Random Forest model was Accuracy 0.995, compared to 0.739 for Decision Tree model. Our Test data set comprises 20 cases. With an accuracy above 99% on our cross-validation data, we can expect that very few, or none, of the test samples will be