Prediction Assignment Writeup

Melanie

28 Juni 2017

Introduction

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).

The project can be found on github: http://benakiva.github.io/practicalML/ and https://github.com/benakiva/practicalMLu can also embed plots, for example:

Source of 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://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Loading and Cleaning the Data

The data sets were saved into workign directory.

```
train <- read.csv('pml-training.csv', na.strings=c("NA",""))
test <- read.csv('pml-training.csv', na.strings=c("NA",""))
```

We remove all columns which have NAs. After this we delete the irrelevant data. In our case irrelevant data are the columns where all values are missing.

```
train<-train[,colSums(is.na(train)) == 0]
test <-test[,colSums(is.na(test)) == 0]

train <- train[,-c(1:7)]
test <- test[,-c(1:7)]
```

Here you see the prepared dataset:

```
## [1] 19622    53

## [1] 19622    53
```

Partition the training dataset

For cross-validation we split our data into a 60% training set and a 40% testing set. For this we are using random subsampling without replacement. Also we use a pseudo-random number generator seed to 1234.

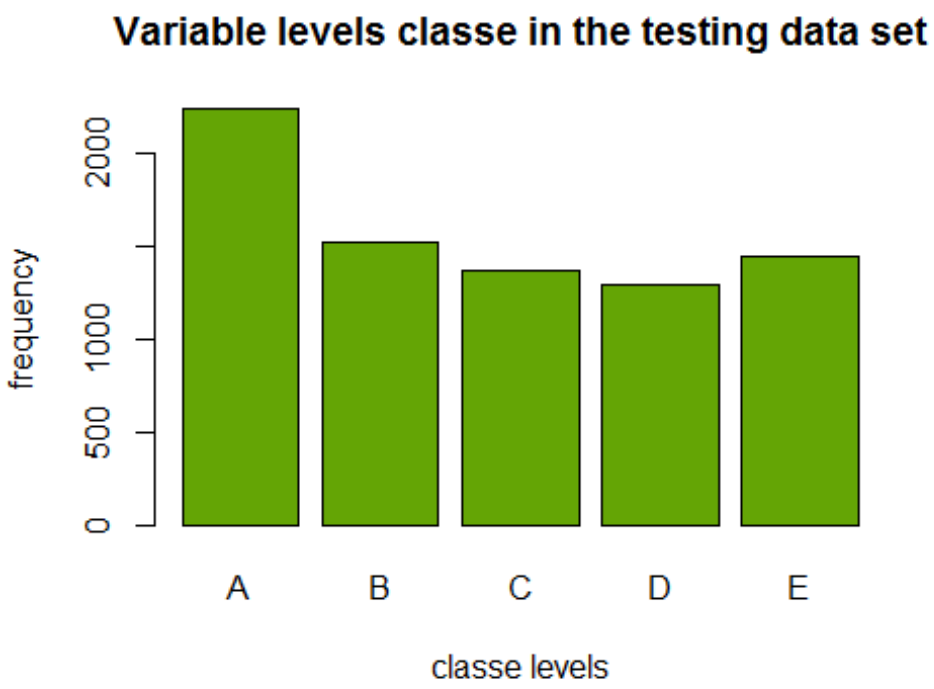
```
set.seed(1234)
xtrain <- createDataPartition(train$classe, p=3/5, list=FALSE)
training <- train[xtrain,]
testing <- train[-xtrain,]
```

Building the model

Data exploration for the classe variable

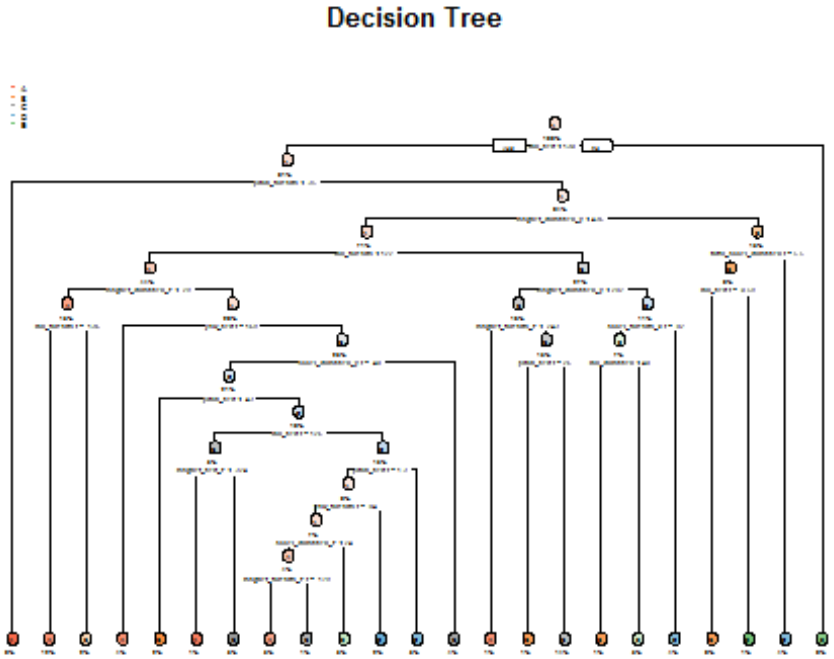
The variable has 5 different levels, as you can see in the plot.

```
plot(testing$classe, col='#64A505', main="Variable levels classe in the testing data set", xlab="classe levels", ylab="frequency")
```



##First model: decision tree We can see a accurancy around 74%. This is less then the second random forest model.

```
dtmodel <- rpart(classe ~ ., data = training, method="class")
rpart.plot(dtmodel, main="Decision Tree", extra=100, under=TRUE, faclen=0)
```



```
#show the result on testing
set.seed(1234)
predicttestdt <- predict(dtmodel, testing, type = "class")
confusionMatrix(predicttestdt, testing$classe)

## Confusion Matrix and Statistics
##
##          Reference
## Prediction    A    B    C    D    E
##          A 1980  212   21   72   31
##          B   85  862   72   90   98
##          C   56  153 1086  209  175
##          D   71  101  110  823   89
##          E   40  190   79   92 1049
##
## Overall Statistics
##
##               Accuracy : 0.7392
##                 95% CI : (0.7294, 0.7489)
##    No Information Rate : 0.2845
##    P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.6699
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                     Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8871   0.5679   0.7939   0.6400   0.7275
## Specificity            0.9401   0.9455   0.9085   0.9434   0.9374
## Pos Pred Value         0.8549   0.7142   0.6468   0.6893   0.7234
## Neg Pred Value         0.9544   0.9012   0.9543   0.9304   0.9386
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2524   0.1099   0.1384   0.1049   0.1337
## Detection Prevalence   0.2952   0.1538   0.2140   0.1522   0.1848
## Balanced Accuracy      0.9136   0.7567   0.8512   0.7917   0.8324
```

Second model: random forest

As we expect you can see a higher accurancy with the Random Forest model then with the decision tree. We get a accurancy around 100%, so the second model is choosen for the coursera quiz.

```
set.seed(1234)
rfmodel <- randomForest(classe ~. , data=training, ntree = 1000)
predicttestrf <- predict(rfmodel, testing, type = "class")

#show the result on testing
confusionMatrix(predicttestrf, testing$classe)

## Confusion Matrix and Statistics
##
##          Reference
```

```
## Prediction    A    B    C    D    E
##          A 2231   11    0    0    0
##          B    0 1502   11    0    0
##          C    0    5 1353   21    2
##          D    1    0    4 1263    2
##          E    0    0    0    2 1438
##
## Overall Statistics
##
##               Accuracy : 0.9925
##                 95% CI : (0.9903, 0.9943)
##    No Information Rate : 0.2845
##    P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.9905
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                     Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9895   0.9890   0.9821   0.9972
## Specificity            0.9980   0.9983   0.9957   0.9989   0.9997
## Pos Pred Value         0.9951   0.9927   0.9797   0.9945   0.9986
## Neg Pred Value         0.9998   0.9975   0.9977   0.9965   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1914   0.1724   0.1610   0.1833
## Detection Prevalence   0.2858   0.1928   0.1760   0.1619   0.1835
## Balanced Accuracy      0.9988   0.9939   0.9924   0.9905   0.9985
```