Assigment

Jaco

25-1-2019

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

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. 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.

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Importing Data:

```
setwd("/Users/jaco/Documents/datasciencecoursera/Course 8_Machine
Learning/Week 4")
dataset <- read.csv('./pml-training.csv', header=T)
test<- read.csv('./pml-testing.csv', header=T)
dim(dataset)
```

```
## [1] 19622   160
```

Exploration and preprocessing:

Above we see that there are 160 variables counting 19622 records. Let's explore te quality of the data:

```
na_count <-sapply(dataset, function(y) sum(length(which(is.na(y)))))
na_count_df <- data.frame(na_count>0)
sum(na_count_df$na_count)

## [1] 67
```

67 columns have NA's, we will exclude these from the dataset;

```
dataset_clean <- dataset[,which(na_count == 0)]
dim(dataset_clean)

## [1] 19622    93
```

93 columns are left for the analysis, now we will exclude the columns with zero or near zero variance:

```
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

NZV <- nearZeroVar(dataset_clean)
dataset_clean <- dataset_clean[-NZV]
dim(dataset_clean)

## [1] 19622    59
```

Finally we exclude the first 6 descriptive columns that can'st be predicters (name / timestamps / num_window):

```
dataset_clean <- dataset_clean[,7:59]
dim(dataset_clean)

## [1] 19622    53
```

53 columns / potential predicters are left to train our machine learning model, we split this cleanes dataset in a training (80%) and validation set (20%) for performing the cross validation:

```
inTrain <- createDataPartition(y=dataset_clean$classe, p=0.8, list=FALSE)
Training <- dataset_clean[inTrain, ]
Validation <- dataset_clean[-inTrain, ]
```

Ok, we're finished the preprocessing proces, the datasets can now be used for building ML models

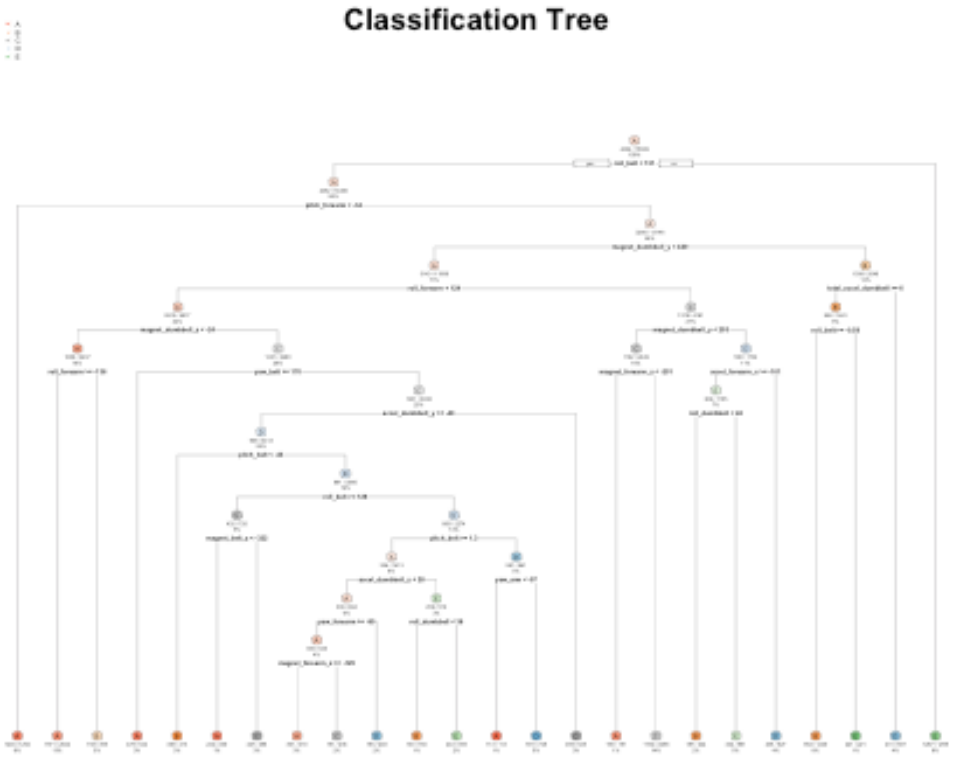
Machine Learning models

We will predict the classification "classe", where we try 2 algorithmns we learned in the course: the Decision Tree and Random Forest Algorithm. We first have a look how the decision tree works out:

```
library(rpart)
library(rpart.plot)

#Classification Tree
set.seed(1234)
mod_dt <- rpart(classe ~ ., data=Training, method="class")

#Plot Decision Tree
rpart.plot(mod_dt, main="Classification Tree", extra=102, under=TRUE,
faclen=0)
```




```
pred_dt_train <- predict(mod_dt, Training, type = "class")
CM_dt <- confusionMatrix(Training$classe, pred_dt_train)
CM_dt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4086  125  124   56   73
##          B  453 1765  409  200  211
##          C   52  138 2229  153  166
##          D  143  199  386 1650  195
##          E   48  225  367  151 2095
##
## Overall Statistics
##
##                Accuracy : 0.7532
##                  95% CI : (0.7464, 0.76)
##     No Information Rate : 0.3046
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6872
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8545   0.7198   0.6341   0.7466   0.7646
## Specificity            0.9654   0.9039   0.9582   0.9316   0.9390
## Pos Pred Value         0.9153   0.5810   0.8141   0.6413   0.7259
## Neg Pred Value         0.9381   0.9457   0.9008   0.9573   0.9497
## Prevalence             0.3046   0.1562   0.2239   0.1408   0.1745
## Detection Rate         0.2603   0.1124   0.1420   0.1051   0.1334
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9099   0.8119   0.7962   0.8391   0.8518

Accuracy_dt <- round(CM_dt$overall[1]*100,2)
```

Results of using the Decision Tree algorithm shows a accuracy of 75.32%. Let's try the Random Forest algorithm:

```
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

mod_rf <- randomForest(classe ~ ., data = Training)
```

```
pred_rf_train <- predict(mod_rf, Training)
CM_rf <- confusionMatrix(Training$classe, pred_rf_train)
CM_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4464    0    0    0    0
##          B    0 3038    0    0    0
##          C    0    0 2738    0    0
##          D    0    0    0 2573    0
##          E    0    0    0    0 2886
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 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   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
Accuracy_rf <- round(CM_rf$overall[1]*100,2)
```

Results of using the Random Forest algorithm shows a accuracy of 100%. Let's validate this model on validation dataset we just created:

Cross Validation of the Random Forest model:

```
pred_rf_val <- predict(mod_rf, Validation)
CM_rf_val <- confusionMatrix(Validation$classe, pred_rf_val)
CM_rf_val
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
```

```
##          B    1  758    0    0    0
##          C    0    3  681    0    0
##          D    0    0    4  639    0
##          E    0    0    2    0  719
##
## Overall Statistics
##
##                Accuracy : 0.9975
##                  95% CI : (0.9953, 0.9988)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9961   0.9913   1.0000   1.0000
## Specificity            1.0000   0.9997   0.9991   0.9988   0.9994
## Pos Pred Value         1.0000   0.9987   0.9956   0.9938   0.9972
## Neg Pred Value         0.9996   0.9991   0.9981   1.0000   1.0000
## Prevalence             0.2847   0.1940   0.1751   0.1629   0.1833
## Detection Rate         0.2845   0.1932   0.1736   0.1629   0.1833
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9996   0.9979   0.9952   0.9994   0.9997
```

```
Accuracy_rf_val <- round(CM_rf_val$overall[1]*100,2)
```

Cross Validation results of the trained Random Forest model shows a accuracy of 99.75%. (Out-of-Sample error is 0.25%).

Conclusion:

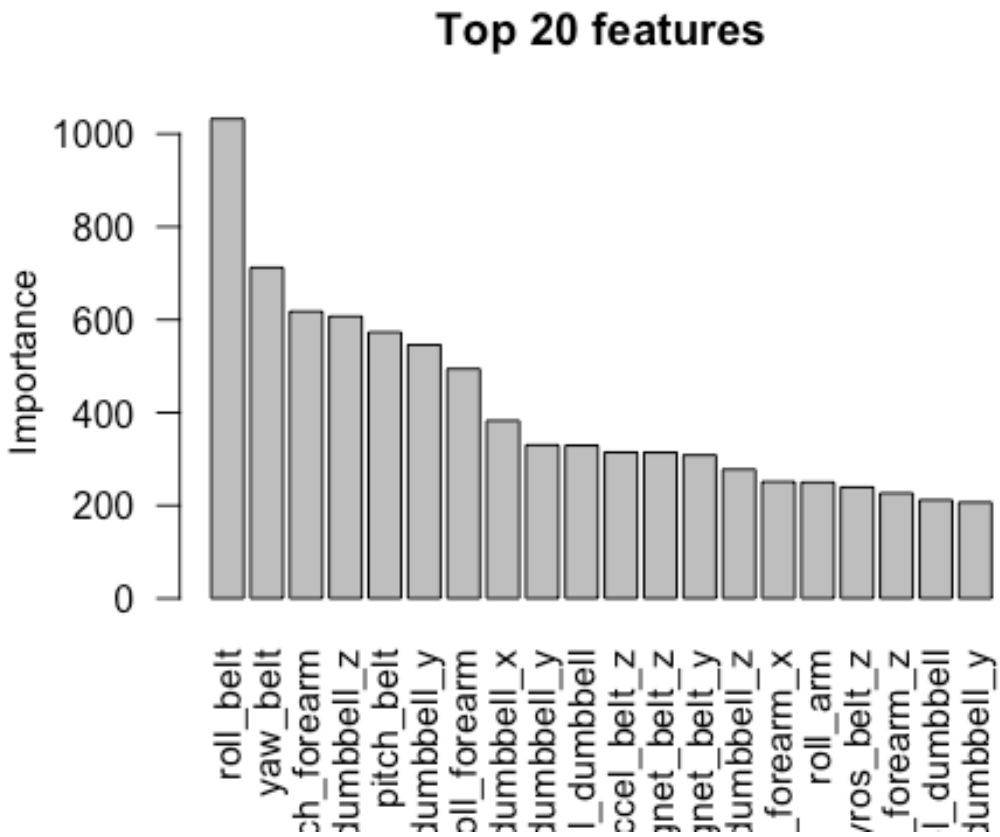
In this case the random forest algorithm outperformed the decision tree algoritm. We were able to predict the "classe" with a high accuracy of 99.75%. The 20 most important features are:

```
Feature_imp <- importance(mod_rf)
Feature_imp <- as.data.frame(as.table(Feature_imp))

Feature_imp$Var2 <- NULL
names(Feature_imp) <- c("Feature", "Importance")

Feature_imp <- head(Feature_imp[order(-Feature_imp$Importance),],20)

barplot(Feature_imp$Importance, names = Feature_imp$Feature,
        xlab = "", ylab = "Importance",
        main = "Top 20 features",
        las=2)
```



Finally we run the model on the test set in order to answer the assignment questions

```
predict_FINAL <- predict(mod_rf, test)
print(predict_FINAL)
```

```
##  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
```