Prediction Assignment WriteUp

m_denboeft 13-4-2018

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, the goal was to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and to predict the manner in which they did the exercise (the “classe” variable in the training set). In this report I descrobe how I build my model, how I used cross validation and what I think the expected out of sample error is and why I made the choices I did. Also I used my prediction model to predict 20 different test cases.

First step: for this assignment the following packages were installed and loaded with install.packages() and library()

```
library(lattice)
library(ggplot2)
library(knitr)
library(caret)

## Warning: package 'caret' was built under R version 3.4.4

library(rpart)
library(rpart.plot)
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.4.4

## 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

library(corrplot)

## corrplot 0.84 loaded

library(ranger)

##
## Attaching package: 'ranger'
```

```
## The following object is masked from 'package:randomForest':
## 
##     importance

library(e1071)
```

Loading the data from the website and Loading the data into R

```
TrainingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-train
ing.csv"
TestURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.c
sv"
TrainingData  <- read.csv(url(TrainingURL))
TestData <- read.csv(url(TestURL))
dim(TrainingData)

## [1] 19622   160

dim(TestData)

## [1]  20 160
```

Cleaning the data by removing missing values (NA) (which are variance, mean, SD) and features that are not in the testing set. Also the first 7 features will be removed because they are not numeric of related to the time-series that are not useful.

```
Features <- names(TestData[,colSums(is.na(TestData)) == 0])[8:59]
```

I will only use the features that are used in testing cases.

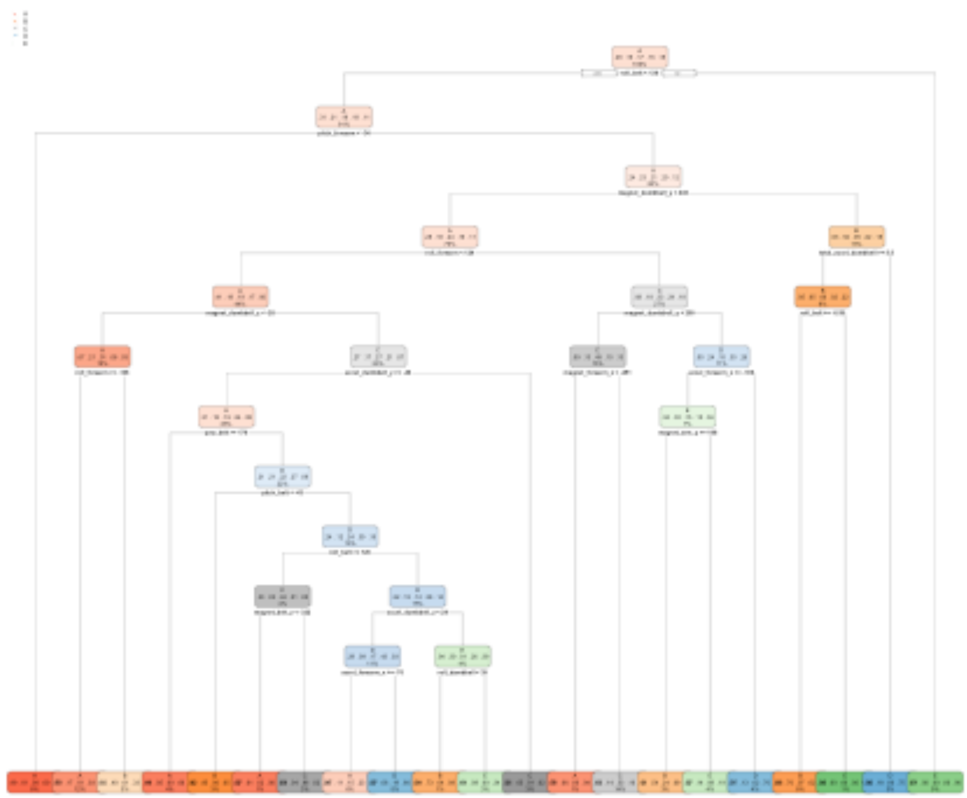
```
TrainingData2 <- TrainingData[,c(Features,"classe")]
TestData2 <- TestData[,c(Features,"problem_id")]
dim(TrainingData2)

## [1] 19622    53

dim(TestData2)

## [1] 20 53
```

Now I will split the data into a training dataset (70% of al cases) and testset (30% of all cases) Hereby I can estimate the out of sample error of the predictor

```
set.seed(12345)
inTrain <- createDataPartition(TrainingData2$classe, p=0.7, list=FALSE)
Training <- TrainingData2[inTrain,]
Testing <- TrainingData2[-inTrain,]
dim(Training)

## [1] 13737    53

dim(Testing)

## [1] 5885   53
```

Building the Decision Tree Model

```
ModelFitDT <- rpart(classe ~ ., data = Training, method="class")
rpart.plot(ModelFitDT)

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



Predicting with the Decision Tree Model

```
set.seed(12345)
Prediction <- predict(ModelFitDT, Testing, type = "class")
confusionMatrix(Prediction, Testing$class)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1498  196   69  106   25
##          B   42  669   85   86   92
##          C   43  136  739  129  131
##          D   33   85   98  553   44
##          E   58   53   35   90  790
##
## Overall Statistics
```

```
##
##                Accuracy : 0.722
##                  95% CI : (0.7104, 0.7334)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6467
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8949   0.5874   0.7203  0.57365   0.7301
## Specificity            0.9060   0.9357   0.9097  0.94717   0.9509
## Pos Pred Value         0.7909   0.6869   0.6273  0.68020   0.7700
## Neg Pred Value         0.9559   0.9043   0.9390  0.91897   0.9399
## Prevalence             0.2845   0.1935   0.1743  0.16381   0.1839
## Detection Rate         0.2545   0.1137   0.1256  0.09397   0.1342
## Detection Prevalence   0.3218   0.1655   0.2002  0.13815   0.1743
## Balanced Accuracy      0.9004   0.7615   0.8150  0.76041   0.8405
```

Building the Random Forest Model. By using the Random Forest Model, the out of sample error should be small. The error will be estimated using the 30% test dataset

```
set.seed(12345)
ModelFitRF <- randomForest(classe ~ ., data = Training, ntree = 1000)
ModelFitRF
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = Training, ntree = 1000)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.52%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B   15 2638    5    0    0 0.0075244545
## C    0   15 2378    3    0 0.0075125209
## D    0    0   22 2229    1 0.0102131439
## E    0    0    1    7 2517 0.0031683168
```

Decision Tree Predicting on the original Testing Data

```
predictionDT <- predict(ModelFitDT, TestData2, type = "class")
predictionDT
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  C  A  B  D  A  C  D  D  A  A  C  B  C  A  E  E  A  B  B  B
## Levels: A B C D E
```

Random Forest Prediction on the original Testing Data

```
predictionRF <- predict(ModelFitRF, TestData2, type = "class")
predictionRF

##  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
```

Conclusion: The results show that the Random Forest Model is very accurate. When I tested this model on the 20 test cases, they were also all correct.