Prediction Assignment Writeup

Claudio Torres Casanelli

27-01-2021

## 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).

## Analysis of the data

I now start loading the data from the csv files and store them in two variables: training and testing.

training <- read.csv("pml-training.csv", na.strings=c("NA", ""))  
training <- training[,colSums(is.na(training)) == 0]  
training <- training[, -c(1,2,3,4,5,6,7)]  
training$classe <- as.factor(training$classe)  
ncol(training)

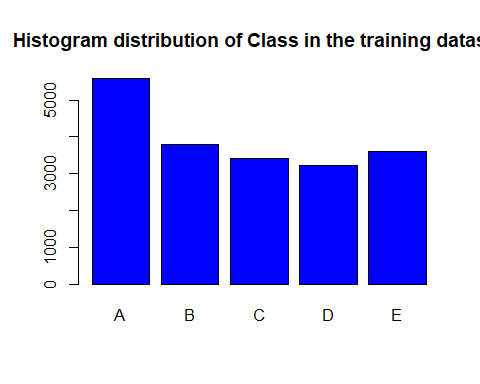
## [1] 53

testing <- read.csv("pml-testing.csv", na.strings=c("NA", ""))  
testing <- testing[,colSums(is.na(testing)) == 0]  
testing <- testing[, -c(1,2,3,4,5,6,7)]  
ncol(testing)

## [1] 53

Cross Validation: We subset the training data set into two and show the proportions of the classe variable for both subsets. The Tree below shows how balanced is the class according to what is on the full training dataset.

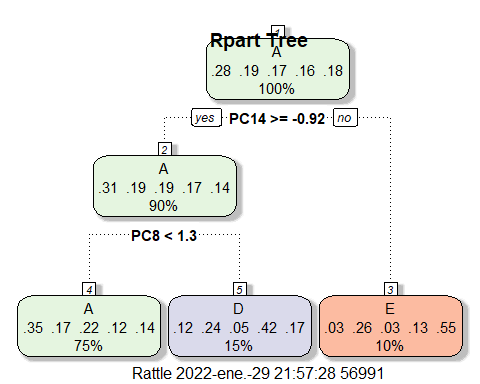
set.seed(23125)  
  
prePCA <- preProcess(training,method="pca",thresh=.95)  
training <- predict(prePCA, training)  
testing <- predict(prePCA, testing)  
  
plot(as.factor(training$classe), main="Histogram distribution of Class in the training dataset", col="blue")



ctrl <- trainControl(method = "cv", number = 5)  
cvmodel <- train(classe~., data = training, method = "rpart", trControl = ctrl)  
cvmodel$finalModel

## n= 19622   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 19622 14042 A (0.28 0.19 0.17 0.16 0.18)   
## 2) PC14>=-0.916215 17652 12134 A (0.31 0.19 0.19 0.17 0.14)   
## 4) PC8< 1.348972 14627 9482 A (0.35 0.17 0.22 0.12 0.14) \*  
## 5) PC8>=1.348972 3025 1767 D (0.12 0.24 0.051 0.42 0.17) \*  
## 3) PC14< -0.916215 1970 887 E (0.031 0.26 0.029 0.13 0.55) \*

fancyRpartPlot(cvmodel$finalModel, main="Rpart Tree")



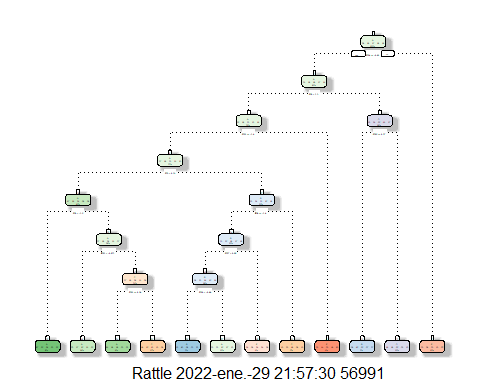
## Fitting a Model

Here I generate models and train the training data set to choose a model to do the final predictions. No need to argue so much about the chosen model. Random Forest is the chosen one. Below is the evidence.

TrainSelect <- createDataPartition(y=training$classe, p=0.8, list=FALSE)  
TrainSelectModel <- training[TrainSelect, ]   
TestSelectModel <- training[-TrainSelect, ]  
  
modelRPART <- rpart(classe~., data=TrainSelectModel, method ="class")  
predRPART <- predict(modelRPART, TestSelectModel, type="class")  
confusionMatrix(predRPART, TestSelectModel$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 832 259 200 170 117  
## B 21 125 8 17 65  
## C 179 133 403 93 108  
## D 61 101 30 242 48  
## E 23 141 43 121 383  
##   
## Overall Statistics  
##   
## Accuracy : 0.506   
## 95% CI : (0.4902, 0.5218)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3665   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.7455 0.16469 0.5892 0.37636 0.53121  
## Specificity 0.7342 0.96492 0.8416 0.92683 0.89756  
## Pos Pred Value 0.5272 0.52966 0.4400 0.50207 0.53868  
## Neg Pred Value 0.8789 0.82804 0.9066 0.88346 0.89477  
## Prevalence 0.2845 0.19347 0.1744 0.16391 0.18379  
## Detection Rate 0.2121 0.03186 0.1027 0.06169 0.09763  
## Detection Prevalence 0.4022 0.06016 0.2335 0.12287 0.18124  
## Balanced Accuracy 0.7399 0.56480 0.7154 0.65160 0.71439

fancyRpartPlot(modelRPART)

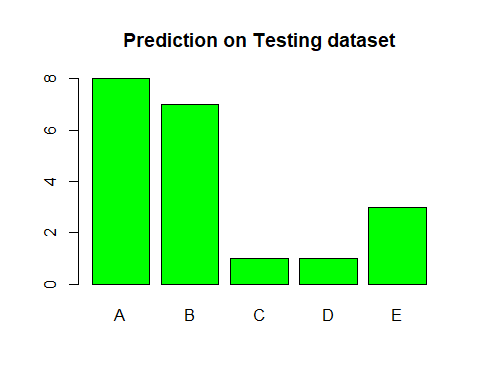


modelRF <- randomForest(classe~., data=TrainSelectModel, method ="class")  
predRF <- predict(modelRF, TestSelectModel, type="class")  
  
confusionMatrix(predRF, TestSelectModel$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1107 15 0 2 0  
## B 2 732 5 0 1  
## C 3 12 671 18 9  
## D 2 0 7 622 2  
## E 2 0 1 1 709  
##   
## Overall Statistics  
##   
## Accuracy : 0.9791   
## 95% CI : (0.9741, 0.9833)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9736   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9919 0.9644 0.9810 0.9673 0.9834  
## Specificity 0.9939 0.9975 0.9870 0.9966 0.9988  
## Pos Pred Value 0.9849 0.9892 0.9411 0.9826 0.9944  
## Neg Pred Value 0.9968 0.9915 0.9960 0.9936 0.9963  
## Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838  
## Detection Rate 0.2822 0.1866 0.1710 0.1586 0.1807  
## Detection Prevalence 0.2865 0.1886 0.1817 0.1614 0.1817  
## Balanced Accuracy 0.9929 0.9809 0.9840 0.9820 0.9911

Finally, now that we have chosen the Random Forest model, I am going to do the prediction on the original Testing data set.

predTesting <- predict(modelRF, testing)  
plot(predTesting, col="green", main="Prediction on Testing dataset")



predTesting

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

## Conclusions

The prediction ressembles the original distribution of the data, so the desition of choosing the Random Forest algorithm was correct from my point of view.