MLearning

mags21

30 de diciembre de 2018

## Getting and cleaning data

library(knitr)

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

library(caret)

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

## Loading required package: lattice

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

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.4.4

library(rpart)  
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.4.4

library(rattle)

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

## Rattle: A free graphical interface for data science with R.  
## Versión 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  
## Escriba 'rattle()' para agitar, sacudir y rotar sus datos.

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:rattle':  
##   
## importance

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

library(corrplot)

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

## corrplot 0.84 loaded

set.seed(12345)  
  
#setwd("C:/Users/MIGUEL GUILLEN/Documents/R")  
setwd("C:/Users/MIGUEL GUILLEN/Documents/GitHub/MachineLearning")  
  
set.seed(1234)  
  
#Load data to memory  
Ttrain <- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!",""))  
Ttest <- read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!",""))

## Partioning the training set into two

# create a partition with the training dataset   
pTrain <- createDataPartition(Ttrain$classe, p=0.7, list=FALSE)  
Train\_Set <- Ttrain[pTrain, ]  
Test\_Set <- Ttrain[-pTrain, ]  
  
dim(Train\_Set)

## [1] 13737 160

dim(Test\_Set)

## [1] 5885 160

## Cleaning the data

# 160 variables. Those variables have plenty of NA  
# The Near Zero variance (NearZV) variables are also removed.  
  
# remove variables with Nearly Zero Variance  
NearZV <- nearZeroVar(Train\_Set)  
Train\_Set <- Train\_Set[, -NearZV]  
Test\_Set <- Test\_Set[, -NearZV]  
dim(Train\_Set)

## [1] 13737 128

dim(Test\_Set)

## [1] 5885 128

# 30 variables removed  
  
  
# remove variables +95% NA  
FullNA <- sapply(Train\_Set, function(x) mean(is.na(x))) > 0.95  
Train\_Set <- Train\_Set[, FullNA==FALSE]  
Test\_Set <- Test\_Set[, FullNA==FALSE]  
dim(Train\_Set)

## [1] 13737 59

dim(Test\_Set)

## [1] 5885 59

# 71 variables removed   
  
# remove identification only variables (columns 1 to 5)  
Train\_Set <- Train\_Set[, -(1:5)]  
Test\_Set <- Test\_Set[, -(1:5)]  
dim(Train\_Set)

## [1] 13737 54

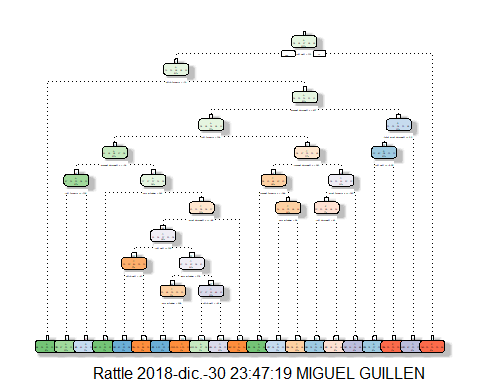
dim(Test\_Set)

## [1] 5885 54

## Using ML algorithms for prediction: Decision Tree

set.seed(12345)  
MDTree <- rpart(classe ~ ., data=Train\_Set, method="class")  
fancyRpartPlot(MDTree)

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



# prediction on Test dataset  
PTree <- predict(MDTree, newdata=Test\_Set, type="class")  
confMatDecTree <- confusionMatrix(PTree, Test\_Set$classe)  
confMatDecTree

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1489 206 54 76 52  
## B 70 671 34 84 111  
## C 19 70 823 128 98  
## D 81 134 49 618 133  
## E 15 58 66 58 688  
##   
## Overall Statistics  
##   
## Accuracy : 0.7288   
## 95% CI : (0.7172, 0.7401)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6557   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.8895 0.5891 0.8021 0.6411 0.6359  
## Specificity 0.9079 0.9370 0.9352 0.9193 0.9590  
## Pos Pred Value 0.7933 0.6918 0.7232 0.6089 0.7774  
## Neg Pred Value 0.9538 0.9048 0.9572 0.9290 0.9212  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2530 0.1140 0.1398 0.1050 0.1169  
## Detection Prevalence 0.3189 0.1648 0.1934 0.1725 0.1504  
## Balanced Accuracy 0.8987 0.7631 0.8687 0.7802 0.7974

confMatDecTree$overall['Accuracy']

## Accuracy   
## 0.728802

## Using ML algorithms for prediction: Random Forest

RF\_C <- trainControl(method="cv", number=3, verboseIter=FALSE)  
MRandForest <- train(classe ~ ., data=Train\_Set, method="rf",  
 trControl=RF\_C)  
MRandForest$finalModel

##   
## Call:  
## randomForest(x = x, y = y, mtry = param$mtry)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 27  
##   
## OOB estimate of error rate: 0.21%  
## Confusion matrix:  
## A B C D E class.error  
## A 3904 1 0 0 1 0.0005120328  
## B 6 2648 3 1 0 0.0037622272  
## C 0 5 2390 1 0 0.0025041736  
## D 0 0 6 2245 1 0.0031083481  
## E 0 1 0 3 2521 0.0015841584

# prediction on Test dataset  
PRandForest <- predict(MRandForest, newdata=Test\_Set)  
confMatRandForest <- confusionMatrix(PRandForest, Test\_Set$classe)  
confMatRandForest

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1674 2 0 0 0  
## B 0 1136 3 0 0  
## C 0 1 1023 1 0  
## D 0 0 0 963 1  
## E 0 0 0 0 1081  
##   
## Overall Statistics  
##   
## Accuracy : 0.9986   
## 95% CI : (0.9973, 0.9994)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9983   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 1.0000 0.9974 0.9971 0.9990 0.9991  
## Specificity 0.9995 0.9994 0.9996 0.9998 1.0000  
## Pos Pred Value 0.9988 0.9974 0.9980 0.9990 1.0000  
## Neg Pred Value 1.0000 0.9994 0.9994 0.9998 0.9998  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2845 0.1930 0.1738 0.1636 0.1837  
## Detection Prevalence 0.2848 0.1935 0.1742 0.1638 0.1837  
## Balanced Accuracy 0.9998 0.9984 0.9983 0.9994 0.9995

confMatRandForest$overall['Accuracy']

## Accuracy   
## 0.9986406

For this case the best model is Random Forest because the accuracy is 0.9964316, betther than Decision Tree.