PMLProject

BY

22/05/2020

library(caret)

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

## Loading required package: ggplot2

library(gbm)

## Warning: package 'gbm' was built under R version 3.6.3

## Loaded gbm 2.1.5

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.6.3

## 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(rpart)

## Warning: package 'rpart' was built under R version 3.6.3

library(e1071)

## Warning: package 'e1071' was built under R version 3.6.3

library(rattle)

## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.

## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.

## Type 'rattle()' to shake, rattle, and roll your data.

##

## Attaching package: 'rattle'

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

##

## importance

Project

It is now possible to collect a large amount of data about personal activity relatively inexpensively Using devices such as Jawbone Up, Nike FuelBand, and Fitbit. 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.

The goal of this project is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The process of building the model, using cross validation, the expected out of sample error, the reasons for the selecting choices are explained in this report.

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 Dataset). The data for this project come from this source: <http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har>.

traind<- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"), header = TRUE)

testd<- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"), header = TRUE)

dim(traind);dim(testd)

## [1] 19622 160

## [1] 20 160

Preparing data

Train dataset includes 19,622 entries and 160 columns. For modeling purpose, the first thing we are going to do is treating missing and outliers values to improve the accuracy of the models. After removing NAs, 93 columns remained. There are also some variables which have no predictve power which were removed. Also variables with no or very little variability in them were removed using nearzerovar function. Finally, there are 53 variables left.

Creating train and test data for prediction

## [1] 13737 53

## [1] 5885 53

Building Model

To predict the outcome, we used 3 methods of Classification tree, Random forest, and boosting.The models were evaluated using cross-validation technique.

1. Classification tree

Building a model with classification tree algorithm, we’ve found an accuracy of about 50% with Out of sample error of 0.5%.

## n= 13737

##

## node), split, n, loss, yval, (yprob)

## \* denotes terminal node

##

## 1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)

## 2) roll\_belt< 129.5 12478 8620 A (0.31 0.21 0.19 0.18 0.11)

## 4) pitch\_forearm< -34 1099 7 A (0.99 0.0064 0 0 0) \*

## 5) pitch\_forearm>=-34 11379 8613 A (0.24 0.23 0.21 0.2 0.12)

## 10) magnet\_dumbbell\_y< 439.5 9626 6914 A (0.28 0.18 0.24 0.19 0.1)

## 20) roll\_forearm< 123.5 6017 3594 A (0.4 0.18 0.19 0.17 0.055) \*

## 21) roll\_forearm>=123.5 3609 2422 C (0.08 0.18 0.33 0.23 0.18) \*

## 11) magnet\_dumbbell\_y>=439.5 1753 853 B (0.031 0.51 0.047 0.23 0.18) \*

## 3) roll\_belt>=129.5 1259 48 E (0.038 0 0 0 0.96) \*

## CART

##

## 13737 samples

## 52 predictor

## 5 classes: 'A', 'B', 'C', 'D', 'E'

##

## No pre-processing

## Resampling: Cross-Validated (5 fold)

## Summary of sample sizes: 10989, 10989, 10989, 10990, 10991

## Resampling results across tuning parameters:

##

## cp Accuracy Kappa

## 0.03631370 0.5164861 0.37325362

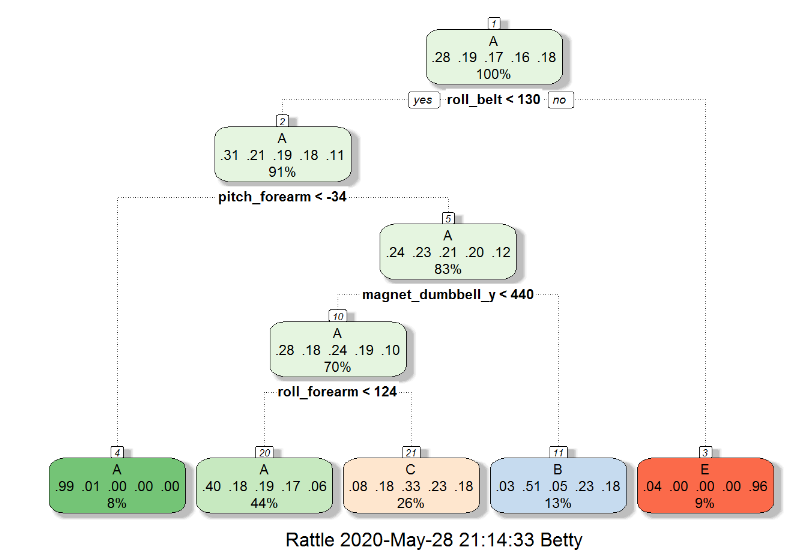
## 0.05913268 0.4179197 0.21166626

## 0.11829926 0.3160848 0.04867898

##

## Accuracy was used to select the optimal model using the largest value.

## The final value used for the model was cp = 0.0363137.



## Confusion Matrix and Statistics

##

## Reference

## Prediction A B C D E

## A 1504 27 117 0 26

## B 470 386 283 0 0

## C 460 26 540 0 0

## D 419 172 373 0 0

## E 158 145 291 0 488

##

## Overall Statistics

##

## Accuracy : 0.4958

## 95% CI : (0.483, 0.5087)

## No Information Rate : 0.5116

## P-Value [Acc > NIR] : 0.9926

##

## Kappa : 0.3418

##

## Mcnemar's Test P-Value : NA

##

## Statistics by Class:

##

## Class: A Class: B Class: C Class: D Class: E

## Sensitivity 0.4995 0.51058 0.33666 NA 0.94942

## Specificity 0.9408 0.85319 0.88648 0.8362 0.88941

## Pos Pred Value 0.8984 0.33889 0.52632 NA 0.45102

## Neg Pred Value 0.6421 0.92204 0.78102 NA 0.99459

## Prevalence 0.5116 0.12846 0.27256 0.0000 0.08734

## Detection Rate 0.2556 0.06559 0.09176 0.0000 0.08292

## Detection Prevalence 0.2845 0.19354 0.17434 0.1638 0.18386

## Balanced Accuracy 0.7202 0.68188 0.61157 NA 0.91941

2. Random forest

Building a model with random forest algorithm, we’ve found an accuracy of about 99.3% with Out of sample error of 0.01.

##

## 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: 2

##

## OOB estimate of error rate: 0.71%

## Confusion matrix:

## A B C D E class.error

## A 3905 1 0 0 0 0.0002560164

## B 17 2636 5 0 0 0.0082768999

## C 0 20 2374 2 0 0.0091819699

## D 0 0 45 2207 0 0.0199822380

## E 0 0 1 6 2518 0.0027722772

## Random Forest

##

## 13737 samples

## 52 predictor

## 5 classes: 'A', 'B', 'C', 'D', 'E'

##

## No pre-processing

## Resampling: Cross-Validated (5 fold)

## Summary of sample sizes: 10989, 10989, 10990, 10990, 10990

## Resampling results across tuning parameters:

##

## mtry Accuracy Kappa

## 2 0.9909733 0.9885804

## 27 0.9903905 0.9878423

## 52 0.9836932 0.9793686

##

## Accuracy was used to select the optimal model using the largest value.

## The final value used for the model was mtry = 2.

## Confusion Matrix and Statistics

##

## Reference

## Prediction A B C D E

## A 1674 0 0 0 0

## B 5 1133 1 0 0

## C 0 3 1023 0 0

## D 0 0 15 948 1

## E 0 0 5 5 1072

##

## Overall Statistics

##

## Accuracy : 0.9941

## 95% CI : (0.9917, 0.9959)

## No Information Rate : 0.2853

## P-Value [Acc > NIR] : < 2.2e-16

##

## Kappa : 0.9925

##

## Mcnemar's Test P-Value : NA

##

## Statistics by Class:

##

## Class: A Class: B Class: C Class: D Class: E

## Sensitivity 0.9970 0.9974 0.9799 0.9948 0.9991

## Specificity 1.0000 0.9987 0.9994 0.9968 0.9979

## Pos Pred Value 1.0000 0.9947 0.9971 0.9834 0.9908

## Neg Pred Value 0.9988 0.9994 0.9957 0.9990 0.9998

## Prevalence 0.2853 0.1930 0.1774 0.1619 0.1823

## Detection Rate 0.2845 0.1925 0.1738 0.1611 0.1822

## Detection Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839

## Balanced Accuracy 0.9985 0.9980 0.9896 0.9958 0.9985

3. Gradient Boosting

Building a model with gradient boosting algorithm, we’ve found an accuracy of about 96.4% with 0.04 Out of sample error.

## A gradient boosted model with multinomial loss function.

## 150 iterations were performed.

## There were 52 predictors of which 52 had non-zero influence.

## Stochastic Gradient Boosting

##

## 13737 samples

## 52 predictor

## 5 classes: 'A', 'B', 'C', 'D', 'E'

##

## No pre-processing

## Resampling: Cross-Validated (5 fold)

## Summary of sample sizes: 10989, 10990, 10990, 10990, 10989

## Resampling results across tuning parameters:

##

## interaction.depth n.trees Accuracy Kappa

## 1 50 0.7506739 0.6838495

## 1 100 0.8196840 0.7717630

## 1 150 0.8528788 0.8138271

## 2 50 0.8546263 0.8157865

## 2 100 0.9064564 0.8816182

## 2 150 0.9316441 0.9135134

## 3 50 0.8982311 0.8711557

## 3 100 0.9420543 0.9266741

## 3 150 0.9600348 0.9494336

##

## Tuning parameter 'shrinkage' was held constant at a value of 0.1

##

## Tuning parameter 'n.minobsinnode' was held constant at a value of 10

## Accuracy was used to select the optimal model using the largest value.

## The final values used for the model were n.trees = 150,

## interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.

## Confusion Matrix and Statistics

##

## Reference

## Prediction A B C D E

## A 1656 11 4 3 0

## B 38 1069 28 4 0

## C 0 27 987 11 1

## D 0 5 36 920 3

## E 5 16 6 21 1034

##

## Overall Statistics

##

## Accuracy : 0.9628

## 95% CI : (0.9576, 0.9675)

## No Information Rate : 0.2887

## P-Value [Acc > NIR] : < 2.2e-16

##

## Kappa : 0.9529

##

## Mcnemar's Test P-Value : 9.842e-12

##

## Statistics by Class:

##

## Class: A Class: B Class: C Class: D Class: E

## Sensitivity 0.9747 0.9477 0.9303 0.9593 0.9961

## Specificity 0.9957 0.9853 0.9919 0.9911 0.9901

## Pos Pred Value 0.9892 0.9385 0.9620 0.9544 0.9556

## Neg Pred Value 0.9898 0.9876 0.9848 0.9921 0.9992

## Prevalence 0.2887 0.1917 0.1803 0.1630 0.1764

## Detection Rate 0.2814 0.1816 0.1677 0.1563 0.1757

## Detection Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839

## Balanced Accuracy 0.9852 0.9665 0.9611 0.9752 0.9931

Final model

We selected Random Forest model for prediction as it has the highest accuracy among all models.

We then applied the model to test data.

## [1] 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