Final Project

Rebeca Riella

23/11/2021

Background

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

Load data

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

```
tes = read_csv("pml-testing.csv")
tr = read_csv("pml-training.csv")
```

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.

```
tr %>% count(classe) %>%
  mutate(Prop = round(n/sum(n)*100,1) ) %>% kable()
```

classe	n	Prop
A	5580	28.4
B	3797	19.4
C	3422	17.4
D	3216	16.4
E	3607	18.4

Data Cleaning

There are many variables with NA. I will remove that. I will remove too the Near Zero Variance(NZR), and the ID variables.

```
# remove columns only contain NA's
tr  = tr [, colSums(is.na(tr)) == 0]
```

```
## Warning: One or more parsing issues, see `problems()` for details
```

```
# remove the Near Zero Variance columns
tr = tr %>% select(- as.vector(nearZeroVar(tr, names=T)))
# remove ID variables
tr <- tr[,-(1:5)]

#remove in testing too
tes = tes %>% select(names(tr %>% select(-classe)))
```

I define training (tr1) and testing (tr2) data.frames

```
aux  = createDataPartition(tr$classe, p=0.7, list=FALSE)
tr1 = tr[aux, ]
tr2  = tr[-aux, ]
```

1) Decision Tree

```
mod = rpart(classe ~ ., data=tr1, method="class")
mod
```

```
## 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< 130.5 12593 8698 A (0.31 0.21 0.19 0.18 0.11)
##        4) pitch_forearm< -33.05 1087   14 A (0.99 0.013 0 0 0) *
##        5) pitch_forearm>=-33.05 11506 8684 A (0.25 0.23 0.21 0.2 0.12)
##         10) num_window>=45.5 11005 8183 A (0.26 0.24 0.22 0.2 0.09)
##           20) magnet_dumbbell_y< 439.5 9398 6634 A (0.29 0.19 0.25 0.19 0.085)
##             40) roll_forearm< 123.5 5902 3414 A (0.42 0.19 0.18 0.16 0.046)
##               80) num_window< 241.5 1476  320 A (0.78 0.11 0.0014 0.067 0.035) *
##               81) num_window>=241.5 4426 3094 A (0.3 0.21 0.25 0.19 0.049)
##                162) magnet_dumbbell_z< -27.5 1371  354 A (0.74 0.19 0.017 0.047 0.0044) *
##                163) magnet_dumbbell_z>=-27.5 3055 1989 C (0.1 0.22 0.35 0.26 0.07)
##                  326) accel_dumbbell_y>=-40.5 2619 1846 D (0.12 0.25 0.25 0.3 0.081)
##                    652) roll_belt>=125.5 628  263 C (0 0.38 0.58 0.038 0.0048)
##                     1304) pitch_belt< -42.75 247   20 B (0 0.92 0 0.069 0.012) *
##                     1305) pitch_belt>=-42.75 381   16 C (0 0.024 0.96 0.018 0) *
##                    653) roll_belt< 125.5 1991 1242 D (0.16 0.21 0.15 0.38 0.11)
##                     1306) yaw_belt< -87.65 821  537 B (0.2 0.35 0.18 0.15 0.12) *
##                     1307) yaw_belt>=-87.65 1170  546 D (0.13 0.11 0.13 0.53 0.092)
##                       2614) pitch_belt< -42.45 352  216 A (0.39 0.33 0.2 0.074 0.011) *
##                       2615) pitch_belt>=-42.45 818  220 D (0.016 0.021 0.11 0.73 0.13) *
##                  327) accel_dumbbell_y< -40.5 436   36 C (0 0.044 0.92 0.039 0) *
##             41) roll_forearm>=123.5 3496 2272 C (0.079 0.19 0.35 0.23 0.15)
##               82) magnet_dumbbell_y< 290.5 2068 1021 C (0.095 0.14 0.51 0.16 0.1)
##                164) num_window< 88.5 169   26 B (0.15 0.85 0 0 0) *
##                165) num_window>=88.5 1899  852 C (0.09 0.08 0.55 0.17 0.11)
##                  330) magnet_dumbbell_z>=284.5 291  142 A (0.51 0.13 0.045 0.12 0.2) *
##                  331) magnet_dumbbell_z< 284.5 1608  574 C (0.014 0.07 0.64 0.18 0.094) *
##               83) magnet_dumbbell_y>=290.5 1428  942 D (0.055 0.26 0.12 0.34 0.22)
##                166) accel_forearm_x>=-101.5 902  607 B (0.049 0.33 0.17 0.14 0.31)
##                  332) roll_dumbbell< 40.19426 176   26 B (0.04 0.85 0.011 0.057 0.04) *
##                  333) roll_dumbbell>=40.19426 726  453 E (0.051 0.2 0.21 0.16 0.38) *
##                167) accel_forearm_x< -101.5 526  168 D (0.067 0.14 0.042 0.68 0.074) *
##           21) magnet_dumbbell_y>=439.5 1607  722 B (0.036 0.55 0.05 0.24 0.12)
##             42) total_accel_dumbbell>=5.5 1116  302 B (0.052 0.73 0.071 0.023 0.12)
##               84) roll_belt>=-0.58 1015  201 B (0.057 0.8 0.078 0.026 0.037) *
##               85) roll_belt< -0.58 101    0 E (0 0 0 0 1) *
##             43) total_accel_dumbbell< 5.5 491  125 D (0 0.14 0.0041 0.75 0.11) *
##         11) num_window< 45.5 501   98 E (0 0 0 0.2 0.8) *
##      3) roll_belt>=130.5 1144   11 E (0.0096 0 0 0 0.99) *
```

Predict Decision Tree

```
pred = predict(mod, tr2 , type = "class")
result = confusionMatrix(pred, factor(tr2$classe))
result
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1526  285   46   86   52
##          B   96  663   91   86   52
##          C    6   60  785  128   62
##          D   26   73   39  546   83
##          E   20   58   65  118  833
## 
## Overall Statistics
##                                           
##                Accuracy : 0.7397          
##                  95% CI : (0.7283, 0.7509)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.6683          
##                                           
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9116   0.5821   0.7651  0.56639   0.7699
## Specificity            0.8886   0.9315   0.9473  0.95509   0.9457
## Pos Pred Value         0.7649   0.6711   0.7541  0.71186   0.7614
## Neg Pred Value         0.9620   0.9028   0.9502  0.91833   0.9480
## Prevalence             0.2845   0.1935   0.1743  0.16381   0.1839
## Detection Rate         0.2593   0.1127   0.1334  0.09278   0.1415
## Detection Prevalence   0.3390   0.1679   0.1769  0.13033   0.1859
## Balanced Accuracy      0.9001   0.7568   0.8562  0.76074   0.8578
```

2) Random Forest

```
set.seed(333)
control2 = trainControl(method="cv", number=3, verboseIter=FALSE)
mod2 = train(classe ~ ., data=tr1, method="rf",
                          trControl=control2)
mod2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.19%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    0    0    0    1 0.0002560164
## B    6 2650    2    0    0 0.0030097818
## C    0    5 2391    0    0 0.0020868114
## D    0    0    8 2243    1 0.0039964476
## E    0    0    0    3 2522 0.0011881188
```

Predict Random Forest

```
predict2 = predict(mod2, newdata=tr2)
result2  = confusionMatrix(predict2, factor(tr2$classe))
result2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    2    0    0    0
##          B    1 1135    0    0    0
##          C    0    2 1026    2    0
##          D    0    0    0  962    3
##          E    0    0    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9969, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9965   1.0000   0.9979   0.9972
## Specificity            0.9995   0.9998   0.9992   0.9994   1.0000
## Pos Pred Value         0.9988   0.9991   0.9961   0.9969   1.0000
## Neg Pred Value         0.9998   0.9992   1.0000   0.9996   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1929   0.1743   0.1635   0.1833
## Detection Prevalence   0.2846   0.1930   0.1750   0.1640   0.1833
## Balanced Accuracy      0.9995   0.9981   0.9996   0.9987   0.9986
```

3) Generalized Boosted Model

```
control3 = trainControl(method = "repeatedcv", number = 5, repeats = 1)
mod3  = train(classe ~ ., data=tr1, method = "gbm",
                    trControl = control3, verbose = FALSE)
mod3$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

Predict Generalized Boosted Model

```
predict3 = predict(mod3, newdata=tr2)
result3  = confusionMatrix(predict3, factor(tr2$classe))
result3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1668    9    0    0    0
##          B    3 1118    2    3    4
##          C    1    9 1017   13    3
##          D    1    3    7  948   14
##          E    1    0    0    0 1061
##
## Overall Statistics
##
##                Accuracy : 0.9876
##                  95% CI : (0.9844, 0.9903)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9843
##
##  Mcnemar's Test P-Value : 0.0002468
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9816   0.9912   0.9834   0.9806
## Specificity            0.9979   0.9975   0.9946   0.9949   0.9998
## Pos Pred Value         0.9946   0.9894   0.9751   0.9743   0.9991
## Neg Pred Value         0.9986   0.9956   0.9981   0.9967   0.9956
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2834   0.1900   0.1728   0.1611   0.1803
## Detection Prevalence   0.2850   0.1920   0.1772   0.1653   0.1805
## Balanced Accuracy      0.9971   0.9895   0.9929   0.9892   0.9902
```

Final Answer

I will chose the best model using the Accuracy: Random Forest

```
result$overall[1]  # D. Tree
```

```
##  Accuracy
## 0.7396771
```

```
result2$overall[1] # R Forest
```

```
##  Accuracy
## 0.9983008
```

```
result3$overall[1] # G. Boosted Model
```

```
##  Accuracy
## 0.9875956
```

Predict classe in Testing database

```
predict(mod2, newdata=tes)
```

```
##  [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
```