Project 4

Facundo Perez

22/4/2020

Resume

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 (see the section on the Weight Lifting Exercise Dataset).

The goal of this work is to predict the manner in which the participants makes the workout. So, we try to predict the classe variable as an outcome. First we clean the data, and split the training dataset into a training and a validation dataset to have a cross validation. With this dataset, we did a tree wich didn't fit well the data. The second model is a model based on bagging. This makes a very good fit. Is very likely that this model (fit 100% in validation) is overfitting, so the out of sample error is very small, but we have to take this with precaution.

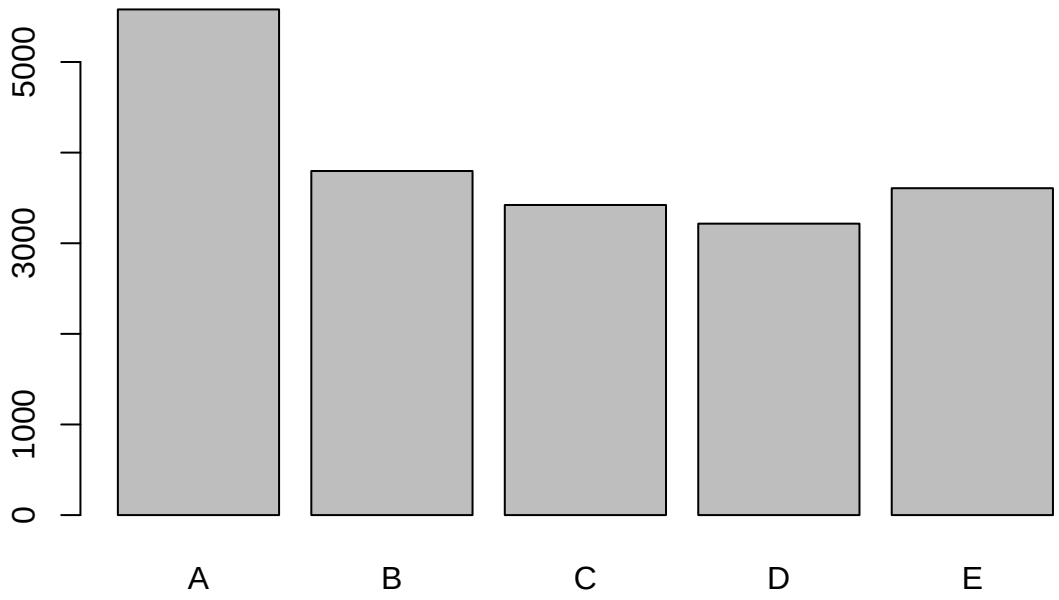
We load libraries, "dplyr", "caret", "rattle", "ggplot2".

Open training and testing sets.

```
training <- read.csv("training.csv")
testing <- read.csv("testing.csv")
```

See the distribution of the variable of interes in the training set.

```
plot(training$classe)
```



Create function for look na and select the variables that doesn't have na.

```
fornas<-function(data){
        nas <- c(rep(0, dim(data)[2]))
                for (i in 1:dim(data)[2]){
                        nas[i] <- sum(is.na(data[,i]))

        }
        nas
}

training <- training[,fornas(training)==0]
testing <- testing[, fornas(testing)==0]
```

Function for eliminate the variables that are empty and eliminate those variables and select the variables of interes.

```
 forempty <-function(data){
         emptys <- c(rep(0, dim(data)[2]))
         for (i in 1:dim(data)[2]){
                 emptys[i] <- sum(data[,i]=='')

         }
         emptys
 }
```

```
training <- training[,forempty(training)==0]
testing <- testing[, forempty(training)==0]
training <- select(training, -X, -new_window, -user_name, -cvtd_timestamp,
                   -raw_timestamp_part_1, -raw_timestamp_part_2, -num_window)
testing <- select(testing, -X, -new_window, -user_name, -cvtd_timestamp,
                   -raw_timestamp_part_1, -raw_timestamp_part_2, -num_window)
```

Now we have a clean dataset.

Cross-validation

We create a partition of the training set call validation.

```
inTrain <- createDataPartition(training$classe, p=0.7, list = FALSE)
training <- training[inTrain,]
validation <-training[-inTrain,]
```

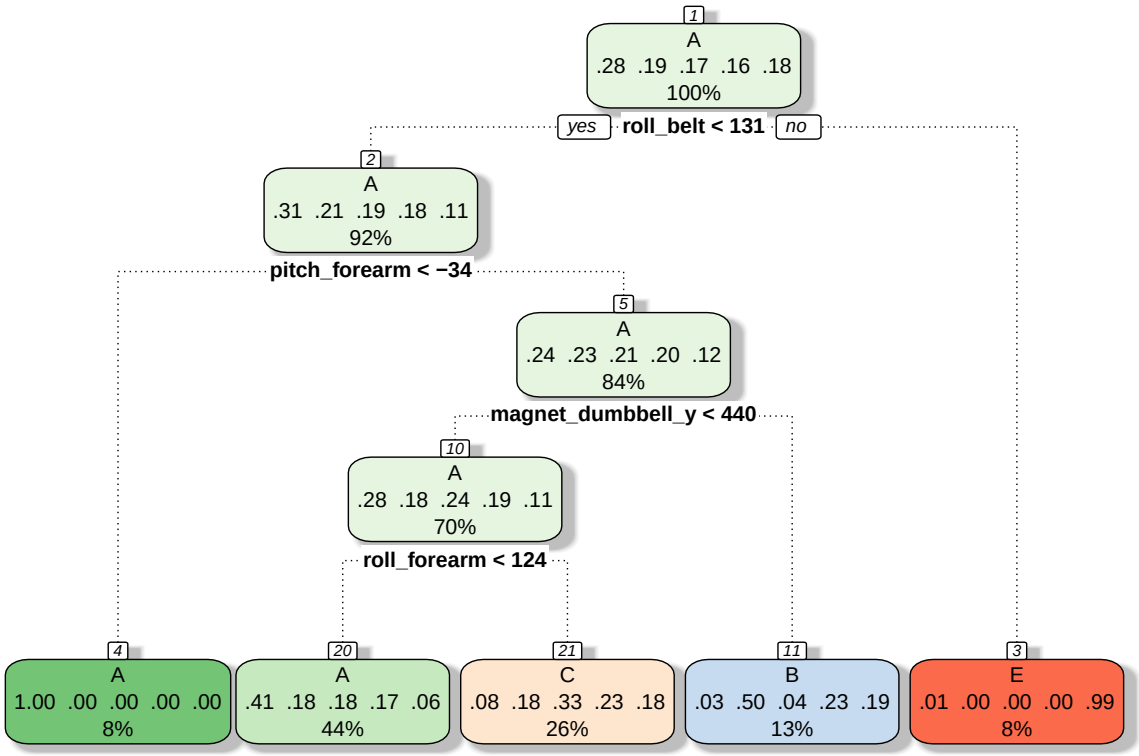
Model 1 Tree

First we apply a tree model. We can see that the accuracy is low. Only 50%.

```
set.seed(7431)
model1 <- train(classe ~ ., data=training, method = "rpart")
model1
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03275353  0.5159358  0.37015187
##   0.05977690  0.4111322  0.20031713
##   0.11484081  0.3396601  0.08400204
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03275353.
```

```
fancyRpartPlot(model1$finalModel)
```



Next, we apply the model to the validation set. We can see that the accuracy is near 50%. The out of sample error is big.

```
predvalidation <- predict(model1, validation)
confusionMatrix(predvalidation, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1074  319  319  298  109
##          B   23  283   28  116  116
##          C   88  170  364  256  209
##          D    0    0    0    0    0
##          E    6    0    0    0  336
##
## Overall Statistics
##
##                Accuracy : 0.5
##                  95% CI : (0.4846, 0.5154)
##     No Information Rate : 0.2895
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3454
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9018  0.36658  0.51195   0.0000  0.43636
## Specificity            0.6425  0.91532  0.78754   1.0000  0.99821
## Pos Pred Value         0.5068  0.50000  0.33487      NaN  0.98246
## Neg Pred Value         0.9414  0.86218  0.88537   0.8371  0.88494
## Prevalence             0.2895  0.18765  0.17282   0.1629  0.18717
## Detection Rate         0.2611  0.06879  0.08848   0.0000  0.08167
## Detection Prevalence   0.5151  0.13758  0.26422   0.0000  0.08313
## Balanced Accuracy      0.7721  0.64095  0.64975   0.5000  0.71728
```

Model 2 Bagging

Next we apply a bagging model. This model have a really good accuracy, near 97%.

```
set.seed(7431)
model2 <- train(classe ~ ., data = training, method = "treebag")
model2
```

```
## Bagged CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9747387  0.9680401
```

We apply the second model to the validation dataset. We can see that the out of sample error is pretty low, because we have a accuracy near 1. Probably we are overfitting.

```
predvalidation2 <- predict(model2, validation)
confusionMatrix(predvalidation2, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1191    0    0    0    0
##          B    0  772    0    0    0
##          C    0    0  711    1    0
##          D    0    0    0  669    0
##          E    0    0    0    0  770
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9998
##                  95% CI : (0.9986, 1)
##     No Information Rate : 0.2895
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9997
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   0.9985   1.0000
## Specificity            1.0000   1.0000   0.9997   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   0.9986   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   0.9997   1.0000
## Prevalence             0.2895   0.1877   0.1728   0.1629   0.1872
## Detection Rate         0.2895   0.1877   0.1728   0.1626   0.1872
## Detection Prevalence   0.2895   0.1877   0.1731   0.1626   0.1872
## Balanced Accuracy      1.0000   1.0000   0.9999   0.9993   1.0000
```

Test set

Last, we apply the bagging model to the test set

```
predtesting <- predict(model2, testing)
```