Practical Machine Learning Project

Fernanda Villeda

14/11/2019

Human Activity Recognition

About the Data

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.

Goal of the project

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions:

- Exactly according to the specification (Class A)
- Throwing the elbows to the front (Class B)
- Lifting the dumbbell only halfway (Class C)
- Lowering the dumbbell only halfway (Class D)
- Throwing the hips to the front (Class E)

Using the variable “classe”, the final model is going to predict the manner in which they did the exercise.

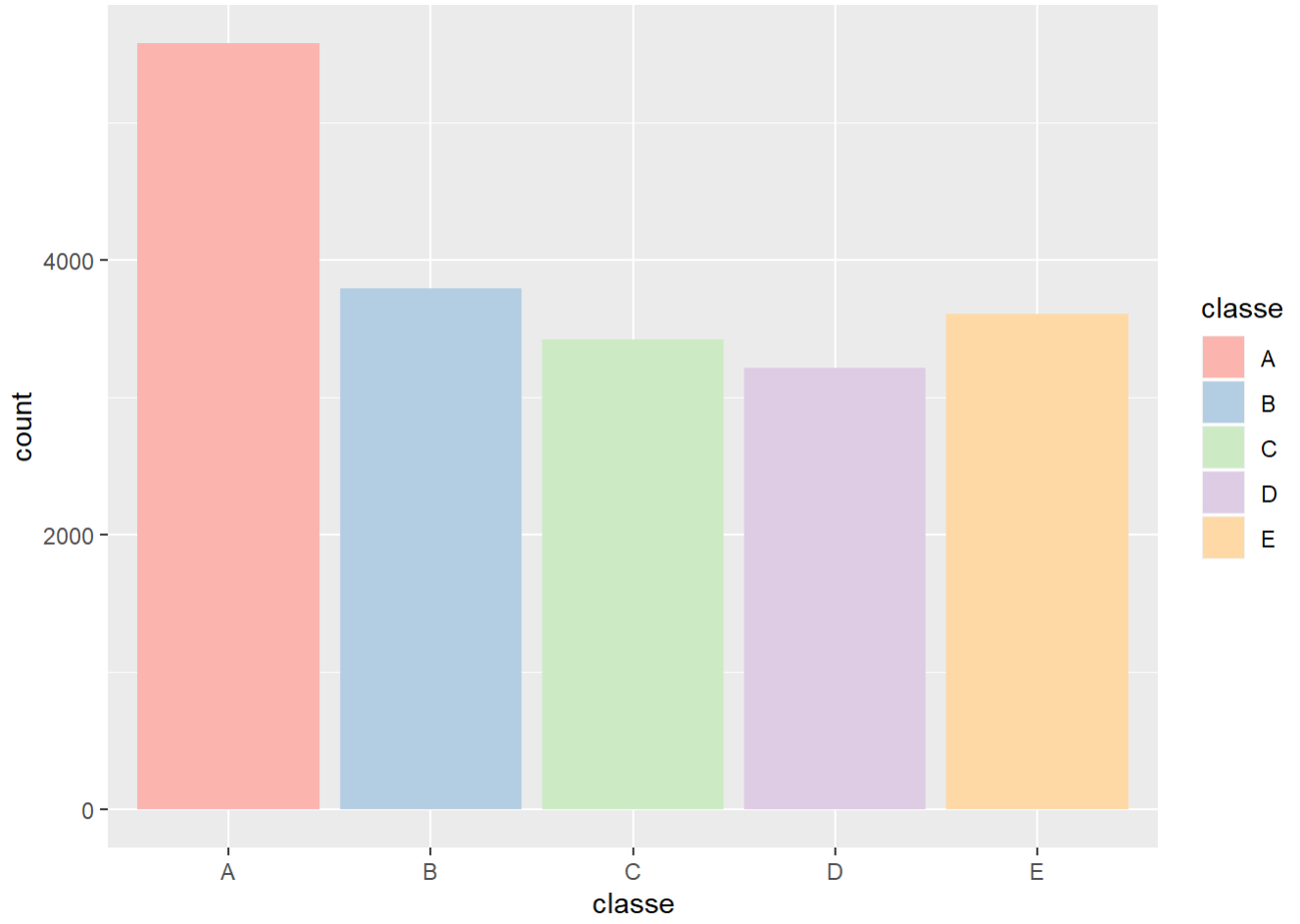
Libraries to use

```
library(readr)
library(caret)
library(rattle)
library(ggplot2)
```

Import and incial analysis

```
pml_testing <- read_csv("pml-testing.csv")
pml_training <- read_csv("pml-training.csv")

prev1 <- ggplot(pml_training, aes(x=classe, fill=classe)) + geom_bar() + scale_fill_brewer(palet
te = "Pastel1")
prev1
```



Train and Test set

I will use the training set. pml_training is separated into two datasets (Trainpml and Trainpml) that will be used to train and test the model, respectively. In addition, the pml_testing set will be used to validate the model and make the final prediction.

```
#Crear datos de validacion para el analisis del error
set.seed(324)
Validationpml <- pml_testing
inTrain  <- createDataPartition(pml_training$classe, p=0.7, list=FALSE)
Trainpml <- pml_training[inTrain, ]
Testpml <- pml_training[-inTrain, ]
```

Cleaning variables

We delete the first 5 variables (“X1” “user_name”, “raw_timestamp_part_1”, “raw_timestamp_part_2”, “cvtd_timestamp”) since they are not relevant for the analysis. Then we eliminate the variables whose variance is close to 0 and the variables whose values are mostly NA, since these characteristics do not contribute anything to the analysis. At the end of the cleaning we get 54 variables.

```
#quitar variables identificadoras 155
Trainpml <- Trainpml[, -(1:5)]
Testpml  <- Testpml[, -(1:5)]
Validationpml <- Validationpml[,-(1:5)]

#Quitar variables de varianza cercana a 0 122
nearZV <- nearZeroVar(Trainpml)
Trainpml <- Trainpml[, -nearZV]
Testpml  <- Testpml[, -nearZV]
Validationpml <- Validationpml[,-nearZV]

#54 variables
mayorityNA    <- sapply(Trainpml, function(x) mean(is.na(x))) > 0.95
Trainpml <- Trainpml[, mayorityNA==FALSE]
Testpml  <- Testpml[, mayorityNA==FALSE]
Validationpml <- Validationpml[,mayorityNA==FALSE]
```

Building Models

3 types of models will be built. I chose these since they are non-lieal models, use the interaction between variables and are in the top of the performance to make predictions. 1. Decision Tree 2. Random Forest 3. Boosting

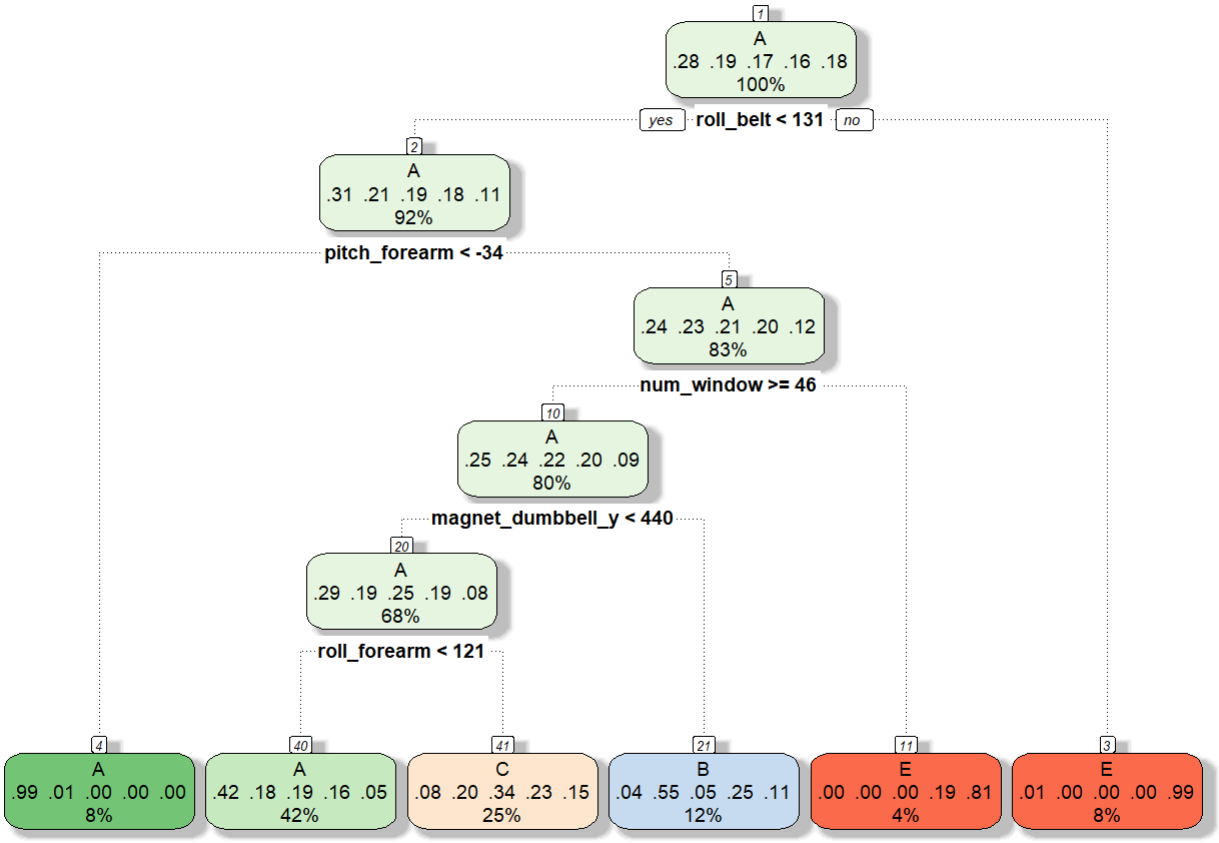
Decision tree

I'll start with the decision tree. using the "rpart" method and the training data set. The resulting tree is shown below.

```
set.seed(324)
modDTree <- train(classe ~., method="rpart", data=Trainpml)
print(modDTree$finalModel)
```

```
## 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 12572 8675 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.55 1141   10 A (0.99 0.0088 0 0 0) *
##      5) pitch_forearm>=-33.55 11431 8665 A (0.24 0.23 0.21 0.2 0.12)
##       10) num_window>=45.5 10935 8169 A (0.25 0.24 0.22 0.2 0.089)
##         20) magnet_dumbbell_y< 439.5 9305 6602 A (0.29 0.19 0.25 0.19 0.085)
##           40) roll_forearm< 120.5 5825 3393 A (0.42 0.18 0.19 0.16 0.047) *
##           41) roll_forearm>=120.5 3480 2287 C (0.078 0.2 0.34 0.23 0.15) *
##         21) magnet_dumbbell_y>=439.5 1630  732 B (0.039 0.55 0.052 0.25 0.11) *
##       11) num_window< 45.5 496   96 E (0 0 0 0.19 0.81) *
##    3) roll_belt>=130.5 1165    9 E (0.0077 0 0 0 0.99) *
```

```
fancyRpartPlot(modDTree$finalModel)
```



Rattle 2019-nov.-19 12:38:09 MariaFernandaVilleda

```
predDtree <- predict(modDTree,newdata=Testpml)
conMDTree <- confusionMatrix(as.factor(Testpml$classe),predDtree)
```

To test the constructed model, I made a confusion matrix using the predictions obtained with the test set and the “classe” column of it. We can see that the performance of the model is not so good, because the precision obtained is 0.5227

```
conMDTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1502   18  149    0    5
##          B  461  388  290    0    0
##          C  465   23  538    0    0
##          D  396  166  352    0   50
##          E   99   93  242    0  648
##
## Overall Statistics
##
##                Accuracy : 0.5227
##                  95% CI : (0.5098, 0.5355)
##     No Information Rate : 0.4967
##     P-Value [Acc > NIR] : 3.5e-05
##
##                   Kappa : 0.3782
##
##  Mcnemar's Test P-Value : < 2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5139  0.56395  0.34246       NA   0.9218
## Specificity            0.9419  0.85549  0.88688   0.8362   0.9162
## Pos Pred Value         0.8973  0.34065  0.52437       NA   0.5989
## Neg Pred Value         0.6626  0.93679  0.78740       NA   0.9885
## Prevalence             0.4967  0.11691  0.26695   0.0000   0.1195
## Detection Rate         0.2552  0.06593  0.09142   0.0000   0.1101
## Detection Prevalence   0.2845  0.19354  0.17434   0.1638   0.1839
## Balanced Accuracy      0.7279  0.70972  0.61467       NA   0.9190
```

Random Forest

To continue building models, I adjusted the following with the random forest method (one of the best in terms of performance) using the Train data set. Then the predictions are made using the test set.

```
#Random Forest
controlRFor <- trainControl(method="cv", number=3, verboseIter=FALSE)
modRFor <- train(classe ~., data=Trainpml, method="rf", trControl=controlRFor)
predRFor <- predict(modRFor,newdata = Testpml)
conMRFor <- confusionMatrix(as.factor(Testpml$classe),predRFor)
#mostrar el arbol elegido
```

The model is shown, which indicates us according to the precision that the tree with the best performance is 27.

```
modRFor
```

```
## Random Forest
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9157, 9159, 9158
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9922836  0.9902387
##   27    0.9967241  0.9958561
##   53    0.9940306  0.9924486
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
modRFor$results[2,]
```

```
##   mtry  Accuracy     Kappa   AccuracySD      KappaSD
## 2   27 0.9967241 0.9958561 0.0007880584 0.0009969609
```

The confusion matrix is shown below. We observe the precision of 0.9988, which indicates that the model responds very well and the predictions are almost exact.

```
conMRFor
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0    1 1025    0    0
##          D    0    0    2  962    0
##          E    0    2    0    2 1078
##
## Overall Statistics
##
##                Accuracy : 0.9988
##                  95% CI : (0.9976, 0.9995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9981   0.9979   1.0000
## Specificity            1.0000   1.0000   0.9998   0.9996   0.9992
## Pos Pred Value         1.0000   1.0000   0.9990   0.9979   0.9963
## Neg Pred Value         1.0000   0.9994   0.9996   0.9996   1.0000
## Prevalence             0.2845   0.1941   0.1745   0.1638   0.1832
## Detection Rate         0.2845   0.1935   0.1742   0.1635   0.1832
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      1.0000   0.9987   0.9989   0.9988   0.9996
```

Boosting

The third model that I will adjust will be using the gbm method corresponding to Boosting. Again, the training is done using the train function and the Trainpml data set, subsequently, the predictions are made using the Testpml data set.

```
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modGBM <- train(classe ~., data=Trainpml, method="gbm", verbose=FALSE, trControl = controlGBM)
predGBM <- predict(modGBM, newdata = Testpml)
conMGBM <- confusionMatrix(as.factor(Testpml$classe),predGBM)
```

the confusion matrix is shown below, again we obtain a significant precision, (0.9908), which indicates that the performance of the model is good.

```
conMGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1667    5    1    1    0
##          B    6 1129    4    0    0
##          C    0    5 1019    0    2
##          D    2    1   12  949    0
##          E    0    2    2   11 1067
##
## Overall Statistics
##
##                Accuracy : 0.9908
##                  95% CI : (0.988, 0.9931)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9884
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9952   0.9886   0.9817   0.9875   0.9981
## Specificity            0.9983   0.9979   0.9986   0.9970   0.9969
## Pos Pred Value         0.9958   0.9912   0.9932   0.9844   0.9861
## Neg Pred Value         0.9981   0.9973   0.9961   0.9976   0.9996
## Prevalence             0.2846   0.1941   0.1764   0.1633   0.1816
## Detection Rate         0.2833   0.1918   0.1732   0.1613   0.1813
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9968   0.9933   0.9901   0.9922   0.9975
```

Comparing Models

We obtain the precision of the 3 models adjusted above (calling the first overall element of each model)

```
conMDTree$overall[1]
```

```
##  Accuracy
## 0.5226848
```

```
conMRFor$overall[1]
```

```
##  Accuracy
## 0.9988105
```

```
conMGBM$overall[1]
```

```
##  Accuracy
## 0.9908241
```

We can notice that the models with better performance are Random forest and Boosting. The result is expected, since both models are usually the most chosen for the precision they obtain. The model with the high level of Accuracy is Random Forest, so we will make the final prediction of the Validationpml data set with this model.

Predicting with the choosen model

We make the prediction of the “class” field of the validationpml data set and assemble the resulting Data frame.

```
#el modelo con mayor accuracy es Random Forest
#aplicar modelo en el conjunto de validacion
predFinal <- predict(modRFor, newdata=Validationpml)
predDF <- data.frame(Validationpml,classe=predFinal)
```

The distribution of the classes is shown in the bar graph below.

```
prev3 <- ggplot(predDF, aes(x=classe, fill=classe)) + geom_bar() + scale_fill_brewer(palette =
"Pastel1")
prev3
```