Prediction Assignment - Course Project

Vinicius Lago

15/09/2020

Introduction

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.
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. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Loading and cleaning data

First, let´s load the two files with our training and testing dataset.

```
# Load training and testing data
training_set <- read.csv("./data/training.csv", na.strings=c("NA","#DIV/0!",""))
testing_set <- read.csv("./data/testing.csv", na.strings=c("NA","#DIV/0!",""))
```

Now, let´s clean our training data, removing the columns that we will not use and the columns with high percent NAs values.

```
# Drop not useful variables
clean <- grep("name|timestamp|window|X", colnames(training_set), value=F)
training_set_clean <- training_set[,-clean]

# Drop variables with high percent NAs (More than 50%)
NArate <- apply(training_set_clean, 2, function(x) sum(is.na(x)))/nrow(training_set_clean)
training_set_clean <- training_set_clean[!(NArate>0.50)]
training_set = training_set_clean
```

The next step consists in split training data in two smaller datasets (75%-25%). The bigger one will be used to train the models and the other one will be used to test the models.

```
# Split training dataset in train and test data
set.seed(123)
ind = sample(2, nrow(training_set), replace = TRUE, prob = c(0.75, 0.25))
train_training_set = training_set[ind ==1, ]
test_training_set = training_set[ind ==2, ]
```

Explore train_training_set data

Let´s see some statistics for train_training_set data.

Our train data contains 53 variables and 14776 observations:

```
dim(train_training_set)
```

```
## [1] 14776    53
```

The main variable of our study (the one we will try to predict) is **classe**. In the next graph we can visualize the distribution of classe in the training data.

```
plot(as.factor(train_training_set$classe),
     col = "red",
     main = "Distribution of classes in train_training_set")
```

Fitting the models

We will fitting three different models to compare and choose the best. We will fitting a model with **Decision tree**, **Ranom Forest** and **GLM**.

To train the models we will use our train_training_set and to validate them we will use the test_training_set. In the moment of training, we will using the method of cross validation with 3 folds to all models.

Decision Tree

First, we will fit the model:

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
model_tree = train(as.factor(classe) ~ .,
                   data = train_training_set,
                   method = "rpart",
                   trControl = trainControl(method = "cv", number = 3))
```

Now, we will predict the test_training_set and analyze the summary statistics.

```r
pred_tree = predict(model_tree, test_training_set)
confusionMatrix(as.factor(test_training_set$classe), as.factor(pred_tree))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1245   24  104    0    5
##          B  377  300  219    0    0
##          C  421   32  431    0    0
##          D  376  131  302    0    0
##          E  149  122  227    0  381
## 
## Overall Statistics
##                                           
##                Accuracy : 0.4864          
##                  95% CI : (0.4722, 0.5006)
##     No Information Rate : 0.5299          
##     P-Value [Acc > NIR] : 1               
##                                           
##                   Kappa : 0.3271          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4848  0.49261  0.33593       NA  0.98705
## Specificity            0.9416  0.85933  0.87286   0.8331  0.88834
## Pos Pred Value         0.9035  0.33482  0.48756       NA  0.43345
## Neg Pred Value         0.6185  0.92177  0.78496       NA  0.99874
## Prevalence             0.5299  0.12567  0.26475   0.0000  0.07965
## Detection Rate         0.2569  0.06191  0.08894   0.0000  0.07862
## Detection Prevalence   0.2844  0.18489  0.18242   0.1669  0.18139
## Balanced Accuracy      0.7132  0.67597  0.60440       NA  0.93769
```

```r
cdt = confusionMatrix(as.factor(test_training_set$classe), as.factor(pred_tree))
```

The accuracy for Decision Tree model is: 0.4863805.

Random Forest

First, we will fit the model:

```
model_rf = train(as.factor(classe) ~ .,
                 method = "rf",
                 data = train_training_set,
                 trControl = trainControl(method = "cv", number = 3))
```

Now, we will predict the test_training_set and analyze the summary statistics.

```
pred_rf = predict(model_rf, test_training_set)
confusionMatrix(as.factor(test_training_set$classe), as.factor(pred_rf))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1378    0    0    0    0
##          B    3  892    1    0    0
##          C    0    2  882    0    0
##          D    0    0   16  791    2
##          E    0    0    3    0  876
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9919, 0.9963)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.993
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9978   0.9778   1.0000   0.9977
## Specificity            1.0000   0.9990   0.9995   0.9956   0.9992
## Pos Pred Value         1.0000   0.9955   0.9977   0.9778   0.9966
## Neg Pred Value         0.9991   0.9995   0.9950   1.0000   0.9995
## Prevalence             0.2850   0.1845   0.1861   0.1632   0.1812
## Detection Rate         0.2844   0.1841   0.1820   0.1632   0.1808
## Detection Prevalence   0.2844   0.1849   0.1824   0.1669   0.1814
## Balanced Accuracy      0.9989   0.9984   0.9887   0.9978   0.9985
```

```r
crf = confusionMatrix(as.factor(test_training_set$classe), as.factor(pred_rf))
```

The accuracy for Random Forest model is: 0.9944284.

GLM

First, we will fit the model:

```r
model_glm = train(as.factor(classe) ~ .,
                  data = train_training_set,
                  trControl = trainControl(method = "cv", number = 3),
                  method='multinom')
```

```
## # weights:  270 (212 variable)
## initial  value 15852.963437
## iter  10 value 13139.339074
## iter  20 value 11573.051679
## iter  30 value 10883.726387
## iter  40 value 10288.644115
## iter  50 value 9711.933214
## iter  60 value 9365.584063
## iter  70 value 9124.482321
## iter  80 value 9034.364328
## iter  90 value 8977.079998
## iter 100 value 8906.645231
## final  value 8906.645231
## stopped after 100 iterations
## # weights:  270 (212 variable)
## initial  value 15852.963437
## iter  10 value 13139.339118
## iter  20 value 11573.051809
## iter  30 value 10883.726793
## iter  40 value 10288.642956
## iter  50 value 9711.933459
## iter  60 value 9365.581316
## iter  70 value 9124.492310
## iter  80 value 9034.378993
## iter  90 value 8977.104794
## iter 100 value 8906.680526
## final  value 8906.680526
## stopped after 100 iterations
## # weights:  270 (212 variable)
## initial  value 15852.963437
## iter  10 value 13139.339074
## iter  20 value 11573.051679
## iter  30 value 10883.726387
```

```
## iter  40 value 10288.644114
## iter  50 value 9711.933215
## iter  60 value 9365.584060
## iter  70 value 9124.482331
## iter  80 value 9034.364343
## iter  90 value 8977.080023
## iter 100 value 8906.645267
## final  value 8906.645267
## stopped after 100 iterations
## # weights:  270 (212 variable)
## initial  value 15854.572875
## iter  10 value 13089.805503
## iter  20 value 11592.203790
## iter  30 value 10778.720270
## iter  40 value 10338.627756
## iter  50 value 9973.326951
## iter  60 value 9755.844073
## iter  70 value 9569.473840
## iter  80 value 9486.349698
## iter  90 value 9436.237098
## iter 100 value 9389.335189
## final  value 9389.335189
## stopped after 100 iterations
## # weights:  270 (212 variable)
## initial  value 15854.572875
## iter  10 value 13089.805532
## iter  20 value 11592.203972
## iter  30 value 10778.720887
## iter  40 value 10338.629269
## iter  50 value 9973.330381
## iter  60 value 9755.850672
## iter  70 value 9569.485952
## iter  80 value 9486.366282
## iter  90 value 9436.259224
## iter 100 value 9389.365796
## final  value 9389.365796
## stopped after 100 iterations
## # weights:  270 (212 variable)
## initial  value 15854.572875
## iter  10 value 13089.805503
## iter  20 value 11592.203790
## iter  30 value 10778.720270
## iter  40 value 10338.627757
## iter  50 value 9973.326954
## iter  60 value 9755.844080
## iter  70 value 9569.473853
## iter  80 value 9486.349714
## iter  90 value 9436.237120
```

```
## iter 100 value 9389.335220
## final  value 9389.335220
## stopped after 100 iterations
## # weights:  270 (212 variable)
## initial  value 15854.572875
## iter  10 value 13278.145430
## iter  20 value 12023.291259
## iter  30 value 11165.160901
## iter  40 value 10736.783036
## iter  50 value 10271.915158
## iter  60 value 10053.814403
## iter  70 value 9859.763667
## iter  80 value 9763.431853
## iter  90 value 9714.111497
## iter 100 value 9645.870844
## final  value 9645.870844
## stopped after 100 iterations
## # weights:  270 (212 variable)
## initial  value 15854.572875
## iter  10 value 13278.145459
## iter  20 value 12023.291433
## iter  30 value 11165.161638
## iter  40 value 10736.784722
## iter  50 value 10271.918541
## iter  60 value 10053.821868
## iter  70 value 9859.776772
## iter  80 value 9763.446974
## iter  90 value 9714.127158
## iter 100 value 9645.903001
## final  value 9645.903001
## stopped after 100 iterations
## # weights:  270 (212 variable)
## initial  value 15854.572875
## iter  10 value 13278.145430
## iter  20 value 12023.291259
## iter  30 value 11165.160901
## iter  40 value 10736.783038
## iter  50 value 10271.915162
## iter  60 value 10053.814410
## iter  70 value 9859.763680
## iter  80 value 9763.431868
## iter  90 value 9714.111513
## iter 100 value 9645.870876
## final  value 9645.870876
## stopped after 100 iterations
## # weights:  270 (212 variable)
## initial  value 23781.054594
## iter  10 value 18838.212214
```

```
## iter  20 value 16755.056878
## iter  30 value 15759.768608
## iter  40 value 14831.643974
## iter  50 value 14287.774234
## iter  60 value 13931.174852
## iter  70 value 13661.982499
## iter  80 value 13528.021183
## iter  90 value 13417.599725
## iter 100 value 13294.297113
## final  value 13294.297113
## stopped after 100 iterations
```

Now, we will predict the test_training_set and analyze the summary statistics.

```
pred_glm = predict(model_glm, test_training_set)
confusionMatrix(as.factor(test_training_set$classe), as.factor(pred_glm))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1174   79   40   62   23
##          B  126  541   78   72   79
##          C  121   87  468  125   83
##          D  100   42   83  528   56
##          E   67  169   25  149  469
##
## Overall Statistics
##
##                Accuracy : 0.6562
##                  95% CI : (0.6426, 0.6696)
##     No Information Rate : 0.3277
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5631
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7393   0.5893  0.67435   0.5641  0.66056
## Specificity            0.9374   0.9096  0.89981   0.9281  0.90087
## Pos Pred Value         0.8520   0.6038  0.52941   0.6527  0.53356
## Neg Pred Value         0.8806   0.9046  0.94296   0.8989  0.93925
## Prevalence             0.3277   0.1894  0.14321   0.1931  0.14651
## Detection Rate         0.2423   0.1116  0.09657   0.1090  0.09678
```

```
## Detection Prevalence   0.2844   0.1849  0.18242   0.1669  0.18139
## Balanced Accuracy      0.8383   0.7495  0.78708   0.7461  0.78072
```

```r
cglm = confusionMatrix(as.factor(test_training_set$classe), as.factor(pred_glm))
```

The accuracy for GLM model is: 0.6562113.

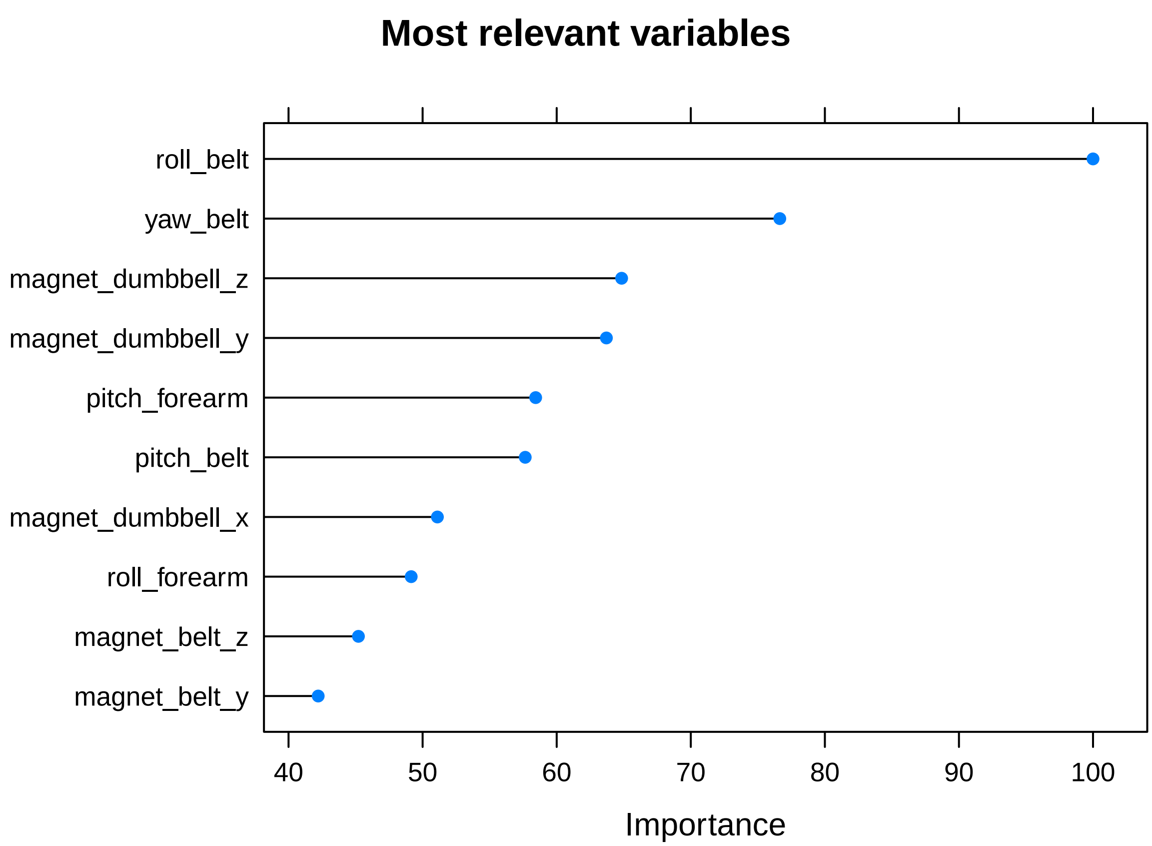
Model Selection

As we can saw, the Random Forest model had the best performance in Accuracy and other measures. So, the Random Forest model was chosen to be our model.

Importance of variables

Before predict the final test set, let´s analyze the importance of variables for our chosen model and get a better understand of it.

```r
# Importance of variables
VarImportance = varImp(model_rf)
plot(VarImportance, main = "Most relevant variables", top = 10)
```



Making prediction for test set

In our last step of this work, let´s make the prediction for test set.

```
# Prediction to testing test
pred_testing_set = predict(model_rf, testing_set)
pred_testing_set
```

```
##  [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
```

The prediction are: B, A, B, A, A, E, D, B, A, A, B, C, B, A, E, E, A, B, B, B.