Practical Machine Learning Peer Assignment

ErickP

16 11 2021

Summary

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 main goal of this assigment is trying to predict if the subject performed an excersice correctly or incorrectly, so this is a binary prediction. Using our knowledge adquired on this course we will create a model that gets the best accuracy.

We will follow what we have learn until now, we divide the project in 5:

```
1)Loading required libraries and data.
2)Exploratory data analysis.
3)Model creation.
4)Model analysis and prediction.
5)Conclusions.
```

1)Loading required libraries and data.

We downloaded the official data from Groupware, they collected the data and made it available for this study.

http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

Then we load it into R-Studio and save it into two variables, Training and Testing.

```
library(caret)
library(ggplot2)
library(rpart)
library(rpart.plot)
library(rattle)
library(randomForest)

Training <- read.csv("pml-training.csv",na.strings = c("NA","#DIV/0!"))
Test <- read.csv("pml-testing.csv",na.strings = c("NA","#DIV/0!"))
```

2)Exploratory data analysis.

After loading the data now we want to do an exploratory data analysis to understand with that type of data we are working with, see the classes of our variables, and start thinking how we will create our model.

```
head(Training)
str(Training)
```

We have 19622 observations of 160 variables, but after exploring more we realize that we have a lot of variables with N/A values, we will remove them from our dataset because they will not be used. We will do the same to the Test dataset.

```
Training <- Filter(function(x) all(!is.na(x)), Training)
Test <- Filter(function(x) all(!is.na(x)), Test)
```

The new data sets have now 60 variables and none of the variables have NAs values. After analyzing the remaining variables we found out that the first 7 variables are not related with the analysis and are more classifiers like the name, time stamps, index of the variable, this ones will not give more information into our model so we decided to remove them too.

```
Training <- Training[,-c(1:7)]
Test <- Test[,-c(1:7)]
```

Now we have 19,622 observations of 53 variables, now we want to see what variables are more correlated to each other. First we need to transform the Training$classe to a factor and this will be the variable we want to predict.

```
Training$classe <- as.factor(Training$classe)
#corr_cross(Training,
  #max_pvalue = 0.05, # display only significant correlations (at 5% level)
  #top = 10 # display top 10 couples of variables (by correlation coefficient)
#)
#Please see Figure 1.1 in the repertory, since I had a problem loading the lares library
#and this grap is done with that library.
```

3) Model Creation.

With this correlation graph, we know which variables are the most correlated and which we will select for our model. We will start by removing all other variables and only focusing in the top 10 of the variables that are correlated.

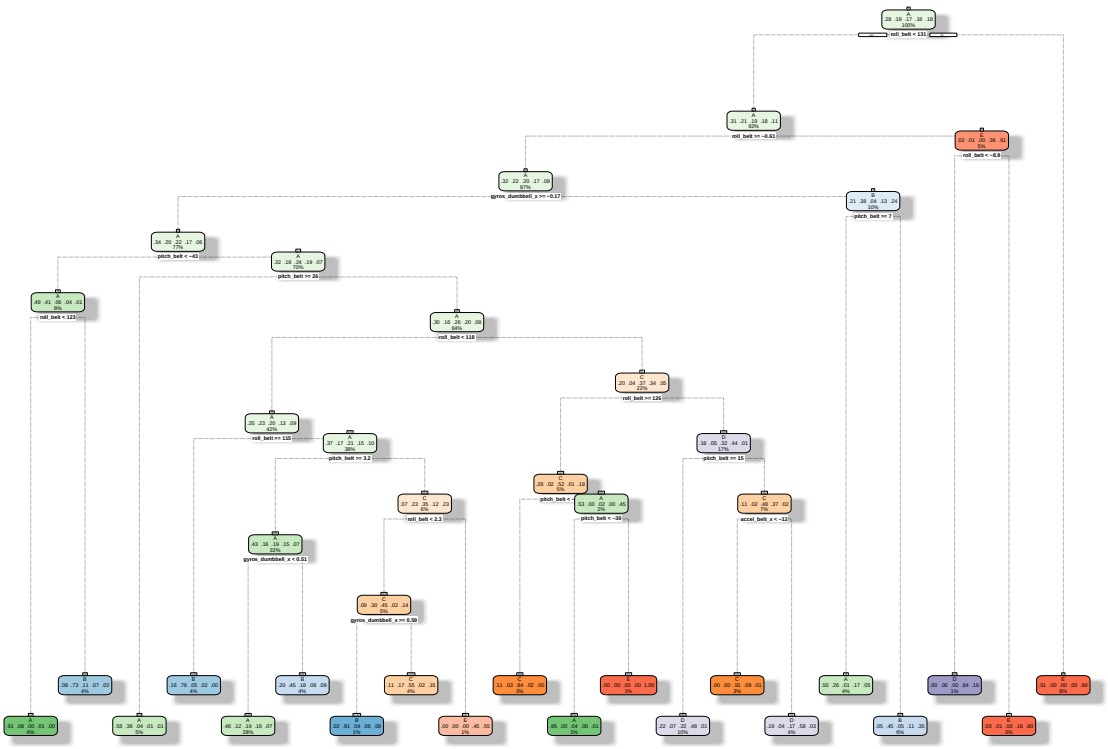
```
set.seed(99999)
modl <- c("classe", "roll_belt", "gyros_forearm_z", "accel_belt_z", "accel_belt_y",
          "accel_belt_x", "total_accel_belt", "gyros_dumbbell_x",
          "gyros_dumbbell_z", "pitch_belt", "gyros_arm_x", "gyros_arm_y")
TrainingM <- Training[,modl]
TestM <- Test[,modl[2:12]]

TPartition  <- createDataPartition(TrainingM$classe, p=0.75, list=FALSE)
TrainM_set <- TrainingM[ TPartition, ]
TestM_set  <- TrainingM[-TPartition, ]
```

4) Model analysis and prediction.

Now that we have our model data sets ready, we can start analyzing and predicting with our model. We start by setting a seed so that our analysis is reproducible, and then we will do a prediction tree.

```
fit_tree <- rpart(classe ~ ., data = TrainM_set, method="class")
fancyRpartPlot(fit_tree)
```



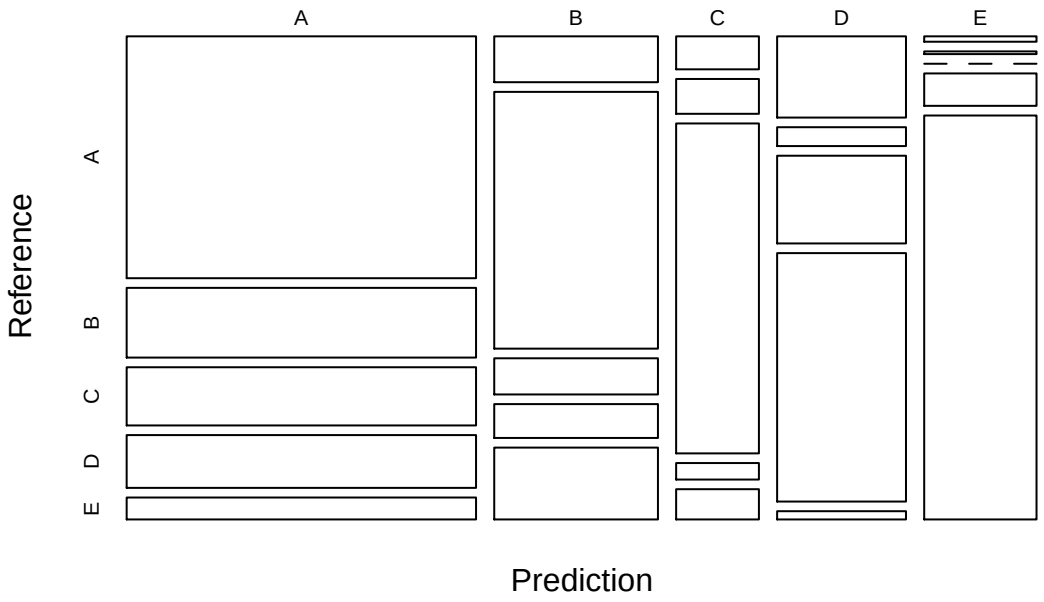
Rattle 2021−Nov−16 12:58:45 rstudio

```
predict_tree <- predict(fit_tree, newdata = TestM_set, type="class")
conf_matrix_decision_tree <- confusionMatrix(predict_tree, factor(TestM_set$classe))
conf_matrix_decision_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1114  321  268  243  101
##          B   99  554   78   73  155
##          C   36   38  360   18   33
##          D  138   32  149  422   14
##          E    8    4    0   48  598
##
## Overall Statistics
##
##                Accuracy : 0.6215
##                  95% CI : (0.6078, 0.6351)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5125
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.7986   0.5838  0.42105  0.52488   0.6637
## Specificity            0.7341   0.8976  0.96913  0.91878   0.9850
## Pos Pred Value         0.5442   0.5777  0.74227  0.55894   0.9088
## Neg Pred Value         0.9016   0.8999  0.88798  0.90793   0.9286
## Prevalence             0.2845   0.1935  0.17435  0.16395   0.1837
## Detection Rate         0.2272   0.1130  0.07341  0.08605   0.1219
## Detection Prevalence   0.4174   0.1956  0.09890  0.15396   0.1342
## Balanced Accuracy      0.7663   0.7407  0.69509  0.72183   0.8244
```

```
plot(conf_matrix_decision_tree$table, col = conf_matrix_decision_tree$byClass,
     main = paste("Decision Tree Model: Predictive Accuracy =",
                  round(conf_matrix_decision_tree$overall['Accuracy'], 4)))
```



Our prediction Accuracy is 0.6215 with this model, which is relative small, but still better than a coin toss. Now we will try to do another type of prediction using Random Forest Model.

```
set.seed(99999)
ctrl_RF <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
fit_RF  <- train(classe ~ ., data = TrainM_set, method = "rf",
                 trControl = ctrl_RF, verbose = FALSE)
fit_RF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)), verbose = FALSE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 6.75%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3983   58   59   79    6  0.04826762
## B   78 2661   69   31    9  0.06566011
```

```
## C   83   65 2304  110    5  0.10245423
## D   83   25  114 2175   15  0.09825871
## E   55   13   19   18 2601  0.03880266
```

```
predict_RF <- predict(fit_RF, newdata = TestM_set)
conf_matrix_RF <- confusionMatrix(predict_RF, factor(TestM_set$classe))
conf_matrix_RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1314   25   31   24   18
##          B   28  886   15    9    7
##          C   24   20  775   38   11
##          D   27   12   34  729    6
##          E    2    6    0    4  859
##
## Overall Statistics
##
##                Accuracy : 0.9305
##                  95% CI : (0.923, 0.9374)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.912
##
##  Mcnemar's Test P-Value : 0.002721
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9419   0.9336   0.9064   0.9067   0.9534
## Specificity            0.9721   0.9851   0.9770   0.9807   0.9970
## Pos Pred Value         0.9306   0.9376   0.8929   0.9022   0.9862
## Neg Pred Value         0.9768   0.9841   0.9802   0.9817   0.9896
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2679   0.1807   0.1580   0.1487   0.1752
## Detection Prevalence   0.2879   0.1927   0.1770   0.1648   0.1776
## Balanced Accuracy      0.9570   0.9593   0.9417   0.9437   0.9752
```

Using the Random Forest Model we get a 93.05 Accuracy, which is way better and the regular Decision Tree Model with 62.15% accuracy.

5) Conclusion.

After analyzing our model we decided to use Random Forest Model for predicting our Test set. We calculated a 93.05% accuracy using this model, so we can be confident that we will get a good result, maybe not the best model out there, but it is a good learning model. Now we will see the results of our model with the Test data.

```
ResultsTestM <- as.data.frame(predict(fit_RF, newdata = TestM))
ResultsTestM
```

```
##    predict(fit_RF, newdata = TestM)
## 1                                 B
## 2                                 D
```

```
## 3                    B
## 4                    D
## 5                    A
## 6                    E
## 7                    D
## 8                    C
## 9                    A
## 10                   A
## 11                   A
## 12                   C
## 13                   B
## 14                   A
## 15                   E
## 16                   B
## 17                   A
## 18                   B
## 19                   B
## 20                   B
```