Coursera - Practical Machine Learning

Ali Magzari

8/30/2021

Introduction

This project consists of predicting the manner in which participants executed an exercise. The data is collected from accelerometers placed on the belt, forearm, arm, and dumbell of six participants.

For a complete explanation: https://www.coursera.org/learn/practical-machine-learning/supplement/PvInj/course-project-instructions-read-first

Methodology

This part covers the methods used to explore the data and choose the appropriate predictive algorithm.

Acquiring, cleaning and exploring data

Set specific seed for experiment replication purposes.

```
set.seed(500)
```

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(rpart)
library(rpart.plot)
library(rattle)
```

```
## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(e1071)
```

Reading both training and testing data

```
data_train <- read.csv("C:/MY FILES/Rdirectory/pml-training.csv")
data_test <- read.csv("C:/MY FILES/Rdirectory/pml-testing.csv")
```

After looking at the data structure and reading carefully the variables, the following ought to be mentioned:

- The data frame contains 19622 observations of 160 variables.
- The desired variable to predict is named "classe" and takes 5 different values (A-E).
- The first 7 variables have no effect on "classe" and can therefore be omitted.

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

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

```
dim(data_train)
```

```
## [1] 19622   153
```

```
dim(data_test)
```

```
## [1]  20 153
```

Both sets have 153 variables, which is logical. The original sets enclosed 160 variables each. After removing 7 columns, we should be left with 153.

After careful inspection, it came to our attention that a lot of columns contain NA entries.

```
colSums(is.na(data_train))
```

```
##                roll_belt               pitch_belt                 yaw_belt
##                        0                        0                        0
##         total_accel_belt       kurtosis_roll_belt      kurtosis_picth_belt
##                        0                        0                        0
##        kurtosis_yaw_belt       skewness_roll_belt     skewness_roll_belt.1
##                        0                        0                        0
##        skewness_yaw_belt            max_roll_belt           max_picth_belt
##                        0                    19216                    19216
##             max_yaw_belt            min_roll_belt           min_pitch_belt
##                        0                    19216                    19216
##             min_yaw_belt      amplitude_roll_belt     amplitude_pitch_belt
##                        0                    19216                    19216
##       amplitude_yaw_belt     var_total_accel_belt            avg_roll_belt
##                        0                    19216                    19216
##         stddev_roll_belt            var_roll_belt           avg_pitch_belt
##                    19216                    19216                    19216
##        stddev_pitch_belt           var_pitch_belt             avg_yaw_belt
##                    19216                    19216                    19216
```

```
##          stddev_yaw_belt             var_yaw_belt             gyros_belt_x
##                    19216                    19216                        0
##             gyros_belt_y             gyros_belt_z             accel_belt_x
##                        0                        0                        0
##             accel_belt_y             accel_belt_z            magnet_belt_x
##                        0                        0                        0
##            magnet_belt_y            magnet_belt_z                 roll_arm
##                        0                        0                        0
##                pitch_arm                  yaw_arm          total_accel_arm
##                        0                        0                        0
##            var_accel_arm             avg_roll_arm          stddev_roll_arm
##                    19216                    19216                    19216
##             var_roll_arm            avg_pitch_arm         stddev_pitch_arm
##                    19216                    19216                    19216
##            var_pitch_arm              avg_yaw_arm           stddev_yaw_arm
##                    19216                    19216                    19216
##              var_yaw_arm              gyros_arm_x              gyros_arm_y
##                    19216                        0                        0
##              gyros_arm_z              accel_arm_x              accel_arm_y
##                        0                        0                        0
##              accel_arm_z             magnet_arm_x             magnet_arm_y
##                        0                        0                        0
##             magnet_arm_z        kurtosis_roll_arm       kurtosis_picth_arm
##                        0                        0                        0
##         kurtosis_yaw_arm        skewness_roll_arm       skewness_pitch_arm
##                        0                        0                        0
##         skewness_yaw_arm             max_roll_arm            max_picth_arm
##                        0                    19216                    19216
##              max_yaw_arm             min_roll_arm            min_pitch_arm
##                    19216                    19216                    19216
##              min_yaw_arm       amplitude_roll_arm      amplitude_pitch_arm
##                    19216                    19216                    19216
##        amplitude_yaw_arm            roll_dumbbell           pitch_dumbbell
##                    19216                        0                        0
##             yaw_dumbbell   kurtosis_roll_dumbbell  kurtosis_picth_dumbbell
##                        0                        0                        0
##    kurtosis_yaw_dumbbell   skewness_roll_dumbbell  skewness_pitch_dumbbell
##                        0                        0                        0
##    skewness_yaw_dumbbell        max_roll_dumbbell       max_picth_dumbbell
##                        0                    19216                    19216
##         max_yaw_dumbbell        min_roll_dumbbell       min_pitch_dumbbell
##                        0                    19216                    19216
##         min_yaw_dumbbell  amplitude_roll_dumbbell amplitude_pitch_dumbbell
##                        0                    19216                    19216
##   amplitude_yaw_dumbbell     total_accel_dumbbell       var_accel_dumbbell
##                        0                        0                    19216
##        avg_roll_dumbbell     stddev_roll_dumbbell        var_roll_dumbbell
##                    19216                    19216                    19216
##       avg_pitch_dumbbell    stddev_pitch_dumbbell       var_pitch_dumbbell
##                    19216                    19216                    19216
##         avg_yaw_dumbbell      stddev_yaw_dumbbell         var_yaw_dumbbell
##                    19216                    19216                    19216
##         gyros_dumbbell_x         gyros_dumbbell_y         gyros_dumbbell_z
##                        0                        0                        0
```

```
##         accel_dumbbell_x         accel_dumbbell_y         accel_dumbbell_z
##                        0                        0                        0
##        magnet_dumbbell_x        magnet_dumbbell_y        magnet_dumbbell_z
##                        0                        0                        0
##             roll_forearm            pitch_forearm              yaw_forearm
##                        0                        0                        0
##    kurtosis_roll_forearm   kurtosis_picth_forearm     kurtosis_yaw_forearm
##                        0                        0                        0
##    skewness_roll_forearm   skewness_pitch_forearm     skewness_yaw_forearm
##                        0                        0                        0
##         max_roll_forearm        max_picth_forearm          max_yaw_forearm
##                    19216                    19216                        0
##         min_roll_forearm        min_pitch_forearm          min_yaw_forearm
##                    19216                    19216                        0
##   amplitude_roll_forearm  amplitude_pitch_forearm    amplitude_yaw_forearm
##                    19216                    19216                        0
##       total_accel_forearm        var_accel_forearm         avg_roll_forearm
##                        0                    19216                    19216
##      stddev_roll_forearm         var_roll_forearm        avg_pitch_forearm
##                    19216                    19216                    19216
##     stddev_pitch_forearm        var_pitch_forearm          avg_yaw_forearm
##                    19216                    19216                    19216
##       stddev_yaw_forearm          var_yaw_forearm          gyros_forearm_x
##                    19216                    19216                        0
##          gyros_forearm_y          gyros_forearm_z          accel_forearm_x
##                        0                        0                        0
##          accel_forearm_y          accel_forearm_z         magnet_forearm_x
##                        0                        0                        0
##         magnet_forearm_y         magnet_forearm_z                   classe
##                        0                        0                        0
```

Each corrupted column have 19216 NA entries (out of 19622)

The code above removes all NA columns and checks if all NA entries have been deleted.

```r
data_train<- data_train[, colSums(is.na(data_train)) == 0]
```

```r
any(colSums(is.na(data_train)) != 0)
```

```
## [1] FALSE
```

The same code block above should be applied to testing data.

```r
data_test<- data_test[, colSums(is.na(data_test)) == 0]
```

```r
any(colSums(is.na(data_test)) != 0)
```

```
## [1] FALSE
```

At this point, the datasets have the following dimensions:

```
dim(data_train)
```

```
## [1] 19622    86
```

```
dim(data_test)
```

```
## [1] 20 53
```

Preparing data

Data sets may contain predictors which value does not change dramatically across observations. These variables, usually called near-zero variance predictors are not informative and could therefore be deleted.

```
data_train <- data_train[, -nearZeroVar(data_train)]
```

```
dim(data_train)
```

```
## [1] 19622    53
```

The line above shows that 33 variables (86 - 53) were judged to be of near-zero variance, and were naturally deleted.

Time to partition our data(training) into training (70%) and validation (30%) sets.

```
data_part <- createDataPartition(data_train$classe, p = 0.7, list = FALSE)
data_train_p <- data_train[data_part, ]
data_valid_p <- data_train[-data_part, ]
```

```
nrow(data_train_p)
```

```
## [1] 13737
```

```
nrow(data_valid_p)
```

```
## [1] 5885
```

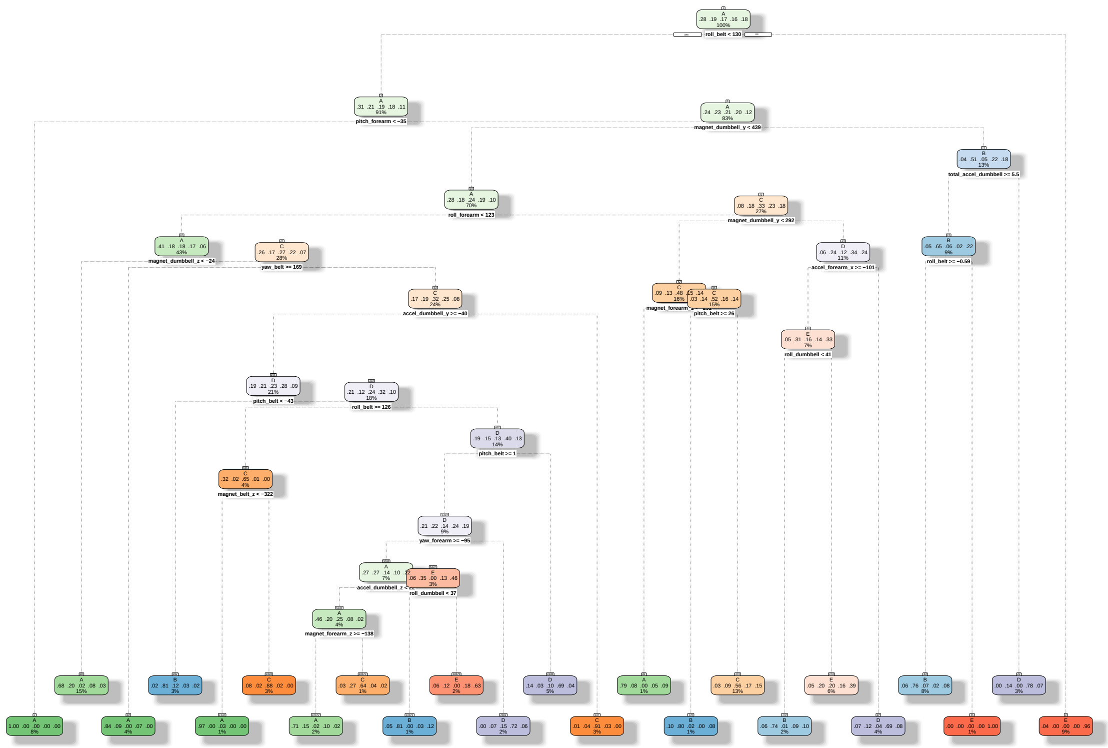
Model building

Two models will be constructed in this part: Decision trees and random forest.

Decision tree

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

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



As shown above, the high number of leaf nodes makes it delicate to visualize the decision tree. Increasing the node size would only cause overlapping.

Let us now use the developed model to predict the variable “class” from the validation set.

```
model_tree_pred <- predict(model_tree, data_valid_p, type = "class")
confMat_tree <- confusionMatrix(model_tree_pred, factor(data_valid_p$classe))
confMat_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1498  227   27  106   37
##          B   37  630   44   21   63
##          C   36  114  835  148  122
##          D   49   92   61  615   57
##          E   54   76   59   74  803
##
## Overall Statistics
##
##                Accuracy : 0.7444
##                  95% CI : (0.7331, 0.7555)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.6755
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8949   0.5531   0.8138   0.6380   0.7421
## Specificity            0.9057   0.9652   0.9136   0.9474   0.9452
## Pos Pred Value         0.7905   0.7925   0.6653   0.7037   0.7533
## Neg Pred Value         0.9559   0.9000   0.9587   0.9304   0.9421
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2545   0.1071   0.1419   0.1045   0.1364
## Detection Prevalence   0.3220   0.1351   0.2133   0.1485   0.1811
## Balanced Accuracy      0.9003   0.7592   0.8637   0.7927   0.8437
```

The confusion matrix displays that the decision tree model was able to predict the class in the validation set with an accuracy of 0.74%, leaving the out-of-sample error to be around 0.26%.

Random forest

This section will cover the construction of a random forest model.

```
control_rf <- trainControl(method = "cv", number = 3, verboseIter = FALSE)
model_rf <- train(classe ~ ., data = data_train_p, method = "rf", trControl = control_rf)
```

Let us now use the developed model to predict the variable "class" from the validation set.

```
model_rf_pred <- predict(model_rf, data_valid_p)
confMat_rf <- confusionMatrix(model_rf_pred, factor(data_valid_p$classe))
confMat_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669    1    0    0    0
##          B    5 1136    2    1    1
##          C    0    2 1021   15    2
##          D    0    0    3  948    5
##          E    0    0    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.9937
##                  95% CI : (0.9913, 0.9956)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.992
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9974   0.9951   0.9834   0.9926
## Specificity            0.9998   0.9981   0.9961   0.9984   1.0000
## Pos Pred Value         0.9994   0.9921   0.9817   0.9916   1.0000
## Neg Pred Value         0.9988   0.9994   0.9990   0.9968   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1930   0.1735   0.1611   0.1825
## Detection Prevalence   0.2838   0.1946   0.1767   0.1624   0.1825
## Balanced Accuracy      0.9984   0.9977   0.9956   0.9909   0.9963
```

From the result above, it is clear that only few entries were predicted wrong (52 to be precise). The accuracy is also reported to be 99%, leaving the out-of-sample error at only 1%.

Conclusion and model implementation

The previous section clearly shows that the random forest algorithm generated a model which accuracy was the highest (99%). It is logical to implement this model to predict the classe on the testing data set.

```
pred_test <- predict(model_rf, data_test)
```

```
pred_test
```

```
##  [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
```

```
table(pred_test)
```

```
## pred_test
## A B C D E
## 7 8 1 1 3
```

References

The data used in this project was obtained from this source: http://groupware.les.inf.puc-rio.br/har