Practical Machine Learning

Edwin Wanner

30 oktober 2018

Overview of project

Devices like 'Jawbone Up', 'Fuelband' and 'Fitbit' collect large amount of data about personal activity. These type of devices are part of the quantified self movement - a group that takes measurements about themselves regularly to improve their health and to find patterns. The goal of this project is to use the data of the devices on the belt, forearm, arm and dumbell of 6 participants to predict the manner in which they did the exercises. In this analysis, three algorithms have been chosen to investigate: Classification trees, Random Forests and Gradient Boosting Models. All of this will be done using cross-validation.

Loading the data

The data is loaded from the following webpages:

- Training data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
- Testing data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

```
trainingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

download.file(trainingURL, destfile = 'training.csv', mode = 'wb')
download.file(testingURL, destfile = 'testing.csv', mode = 'wb')

training <- read.csv(file = "training.csv", header=TRUE, sep = ",")
testing <- read.csv(file = "testing.csv", header = TRUE, sep = "," )
```

Data preprocessing

Looking at the data, the data has a lot of NA and blank values. We will remove those variables which have 95% blank or NA values. These variables add too little value.

```
colSums(is.na(training) | training == "")
```

```
##                        X                user_name     raw_timestamp_part_1
##                        0                        0                        0
##     raw_timestamp_part_2           cvtd_timestamp               new_window
##                        0                        0                        0
##               num_window                roll_belt               pitch_belt
##                        0                        0                        0
##                yaw_belt         total_accel_belt       kurtosis_roll_belt
##                        0                        0                    19216
##      kurtosis_picth_belt        kurtosis_yaw_belt       skewness_roll_belt
##                    19216                    19216                    19216
##     skewness_roll_belt.1        skewness_yaw_belt            max_roll_belt
##                    19216                    19216                    19216
##           max_picth_belt             max_yaw_belt            min_roll_belt
##                    19216                    19216                    19216
##           min_pitch_belt             min_yaw_belt      amplitude_roll_belt
##                    19216                    19216                    19216
```

```
##     amplitude_pitch_belt       amplitude_yaw_belt     var_total_accel_belt
##                    19216                    19216                    19216
##            avg_roll_belt         stddev_roll_belt            var_roll_belt
##                    19216                    19216                    19216
##           avg_pitch_belt        stddev_pitch_belt           var_pitch_belt
##                    19216                    19216                    19216
##             avg_yaw_belt          stddev_yaw_belt             var_yaw_belt
##                    19216                    19216                    19216
##             gyros_belt_x             gyros_belt_y             gyros_belt_z
##                        0                        0                        0
##             accel_belt_x             accel_belt_y             accel_belt_z
##                        0                        0                        0
##            magnet_belt_x            magnet_belt_y            magnet_belt_z
##                        0                        0                        0
##                roll_arm                 pitch_arm                  yaw_arm
##                        0                        0                        0
##          total_accel_arm            var_accel_arm             avg_roll_arm
##                        0                    19216                    19216
##          stddev_roll_arm             var_roll_arm            avg_pitch_arm
##                    19216                    19216                    19216
##         stddev_pitch_arm            var_pitch_arm              avg_yaw_arm
##                    19216                    19216                    19216
##           stddev_yaw_arm             var_yaw_arm              gyros_arm_x
##                    19216                    19216                        0
##              gyros_arm_y              gyros_arm_z              accel_arm_x
##                        0                        0                        0
##              accel_arm_y              accel_arm_z             magnet_arm_x
##                        0                        0                        0
##             magnet_arm_y             magnet_arm_z        kurtosis_roll_arm
##                        0                        0                    19216
##        kurtosis_picth_arm         kurtosis_yaw_arm        skewness_roll_arm
##                    19216                    19216                    19216
##        skewness_pitch_arm         skewness_yaw_arm             max_roll_arm
##                    19216                    19216                    19216
##             max_picth_arm              max_yaw_arm             min_roll_arm
##                    19216                    19216                    19216
##             min_pitch_arm              min_yaw_arm       amplitude_roll_arm
##                    19216                    19216                    19216
##      amplitude_pitch_arm        amplitude_yaw_arm            roll_dumbbell
##                    19216                    19216                        0
##           pitch_dumbbell             yaw_dumbbell   kurtosis_roll_dumbbell
##                        0                        0                    19216
##  kurtosis_picth_dumbbell    kurtosis_yaw_dumbbell   skewness_roll_dumbbell
##                    19216                    19216                    19216
##  skewness_pitch_dumbbell    skewness_yaw_dumbbell        max_roll_dumbbell
##                    19216                    19216                    19216
##        max_picth_dumbbell         max_yaw_dumbbell        min_roll_dumbbell
##                    19216                    19216                    19216
##        min_pitch_dumbbell         min_yaw_dumbbell  amplitude_roll_dumbbell
##                    19216                    19216                    19216
## amplitude_pitch_dumbbell   amplitude_yaw_dumbbell     total_accel_dumbbell
##                    19216                    19216                        0
##        var_accel_dumbbell         avg_roll_dumbbell     stddev_roll_dumbbell
##                    19216                    19216                    19216
```

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

```
blank_ids <- which(apply(training, 2,function(x) sum(x == "")) > 0.95*length(training[,1]))
NA_ids <- which(apply(training, 2,function(x) sum(is.na(x))) > 0.95*length(training[,1]))
trainingCleaned <- training[,-c(blank_ids,NA_ids)]
dim(trainingCleaned)
```

```
## [1] 19622    60
```

```
blank_ids_test <- which(apply(testing, 2,function(x) sum(x == "")) > 0.95*length(testing[,1]))
NA_ids_test <- which(apply(testing, 2,function(x) sum(is.na(x))) > 0.95*length(testing[,1]))
testingCleaned <- testing[,-c(blank_ids_test, NA_ids_test)]
dim(testingCleaned)
```

```
## [1] 20 60
```

Now that we have removed the variables which are blank or NA, we have 60 variables left. Looking at the remaining variables, we see that the first 7 columns are not relevant ('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2', 'cvtd_timestamp', 'new_window','num_window'). These columns are removed.

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

Next, we will create a separate training and test set from the current dataset.

```
set.seed(123)
inTraining <- createDataPartition(trainingCleaned$classe, p = 0.8, list = FALSE)
train_set <- trainingCleaned[inTraining,]
test_set <- trainingCleaned[-inTraining,]
dim(train_set)
```

```
## [1] 15699    53
```

```
dim(test_set)
```

```
## [1] 3923   53
```

Model building

Now that we have got our training- and test set, a number of models will be trained:

1. Classification tree
2. Random forest (multiple classification trees)
3. Gradient Boosting Models

To prevent overfitting, 5-fold cross-validation is applied. The number 5 is chosen as it is frequently chosen when performing cross-validation.

Classification tree

```
control <- trainControl(method = "cv", number = 5, allowParallel = T)
fit_CT <- train(classe~., data = train_set, method = "rpart", trControl = control)
predict_CT <- predict(fit_CT, newdata = test_set)
cm <- confusionMatrix(test_set$classe, predict_CT)
cm$overall[1]
```

```
##  Accuracy
## 0.4973235
```

The classification tree, trained with 5-fold cross-validation, resulted in an accuracy of 49.7%.

Random Forest

Next, we will create a random forest. Again, 5-fold cross-validation is applied.

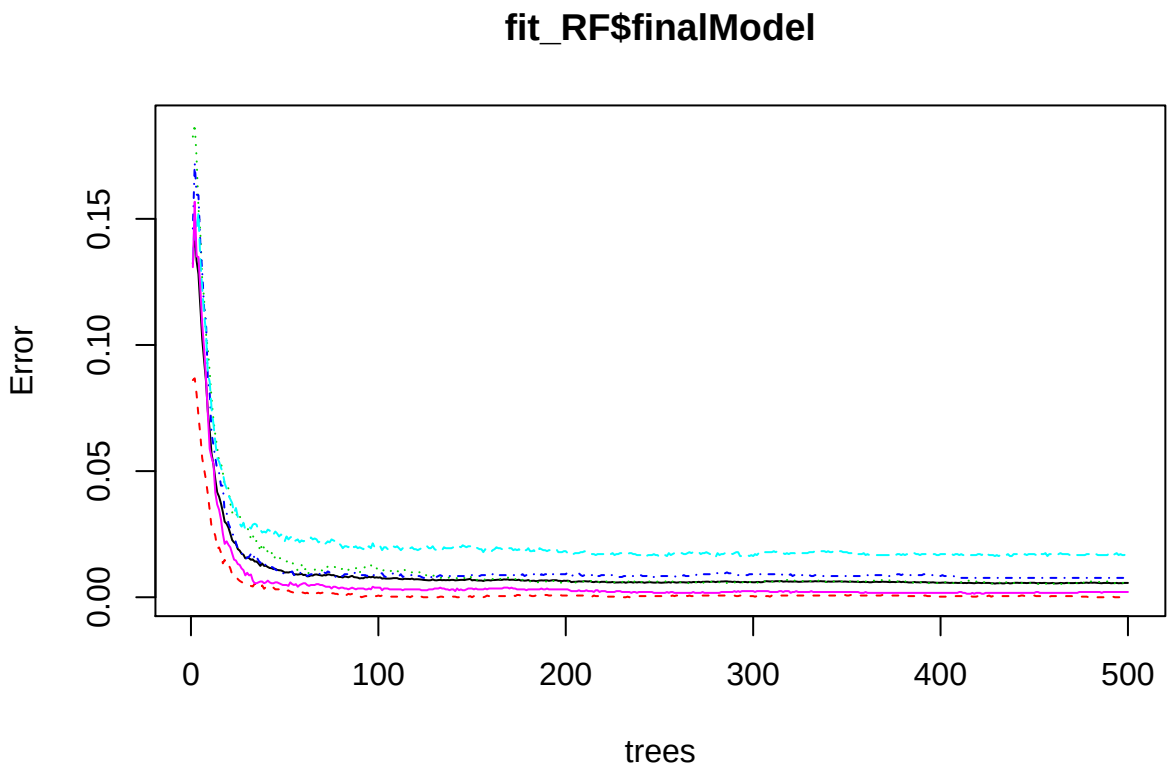
```
fit_RF <- train(classe~., data = train_set, method = "rf", verbose = FALSE,trControl = control)
predict_RF <- predict(fit_RF, newdata = test_set)
cm_RF <- confusionMatrix(test_set$classe, predict_RF)
cm_RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1115    1    0    0    0
##          B    5  749    5    0    0
##          C    0    4  679    1    0
##          D    0    0    8  635    0
```

```
##          E    0    0    0    1  720
##
## Overall Statistics
##
##                Accuracy : 0.9936
##                  95% CI : (0.9906, 0.9959)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9919
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9955   0.9934   0.9812   0.9969   1.0000
## Specificity            0.9996   0.9968   0.9985   0.9976   0.9997
## Pos Pred Value         0.9991   0.9868   0.9927   0.9876   0.9986
## Neg Pred Value         0.9982   0.9984   0.9960   0.9994   1.0000
## Prevalence             0.2855   0.1922   0.1764   0.1624   0.1835
## Detection Rate         0.2842   0.1909   0.1731   0.1619   0.1835
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9976   0.9951   0.9898   0.9972   0.9998
```

Some graphs of the Random Forest show that the error decreases rapidly and stabilizes after creating around 60 trees.

```
plot(fit_RF$finalModel)
```



The Random forest, trained with 5-fold cross-validation, resulted in an accuracy of 99.4% for the test set. This is a major increase in the accuracy compared to the classification tree.

Gradient Boosting

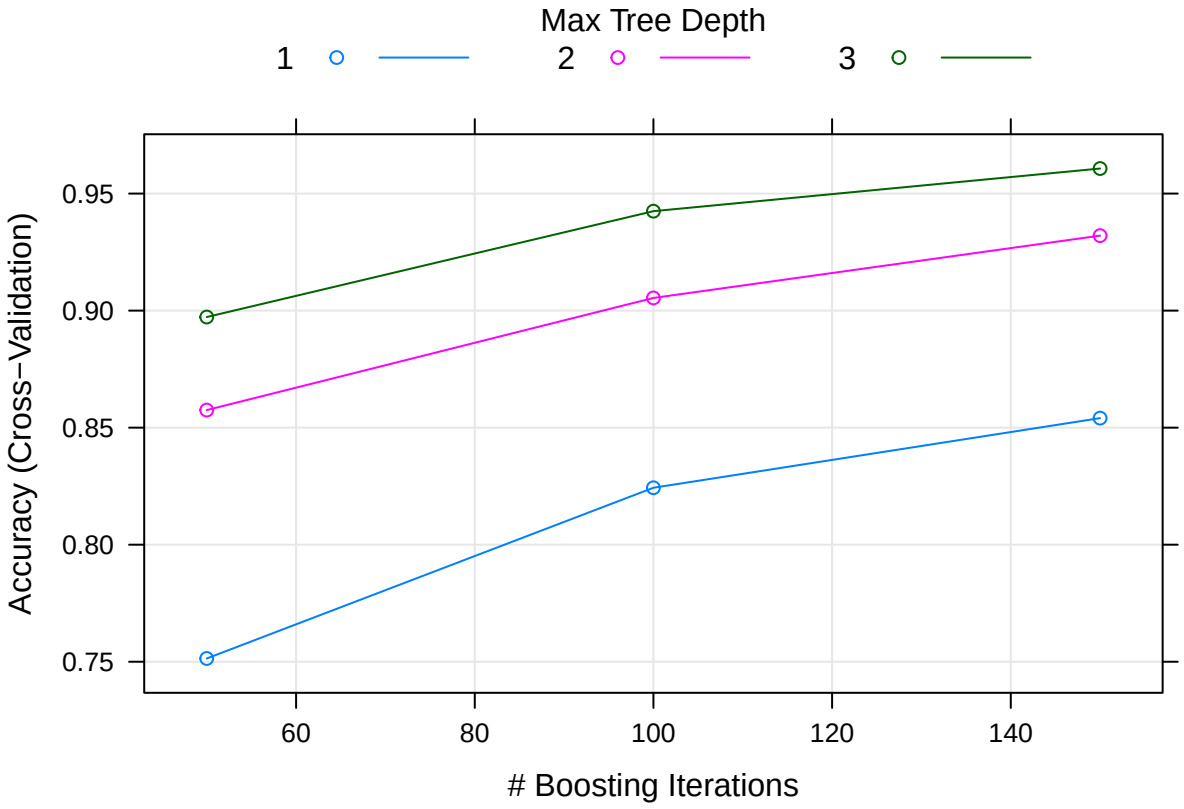
```
fit_GBM <- train(classe~., data = train_set, method = "gbm", verbose = FALSE,trControl = control)
predict_GBM <- predict(fit_GBM, newdata = test_set)
cm_GBM <- confusionMatrix(test_set$classe, predict_GBM)
cm_GBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1097   12    4    3    0
##          B   26  714   17    0    2
##          C    0   18  656    9    1
##          D    0    2   24  612    5
##          E    4    6    3    8  700
##
## Overall Statistics
##
##                Accuracy : 0.9633
##                  95% CI : (0.9569, 0.969)
##     No Information Rate : 0.2873
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9536
##  Mcnemar's Test P-Value : 0.001395
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9734   0.9495   0.9318   0.9684   0.9887
## Specificity            0.9932   0.9858   0.9913   0.9906   0.9935
## Pos Pred Value         0.9830   0.9407   0.9591   0.9518   0.9709
## Neg Pred Value         0.9893   0.9880   0.9852   0.9939   0.9975
## Prevalence             0.2873   0.1917   0.1795   0.1611   0.1805
## Detection Rate         0.2796   0.1820   0.1672   0.1560   0.1784
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9833   0.9676   0.9616   0.9795   0.9911
```

Some plots of the model show that using a higher tree depth provides a higher accuracy. Furthermore, using more boosting iterations could get you an even higher accuracy.

```
plot(fit_GBM)
```



The Gradient Boosting model, trained with 5-fold cross-validation, resulted in an accuracy of 96.5%.

Conclusion

Looking at the results of the three models, one can conclude that the random forest model performed best with an accuracy of 99.4%. The best model is used to predict the outcome of the testing dataset.

```
predict_testing <- predict(fit_RF, newdata = testingCleaned)
predict_testing
```

```
##  [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
```