Predict exercise class

Fabrizio P.

31 July 2016

Summary

In the following analysis a predictive model has been developed in order to predict the quality of a 1.25 dumbell curl. The prediction is based on gyroscopic measurements performed using accelerometers placed on the belt, forearm, arm, and dumbell of 6 participants.

Three models have been tested: random forest, decision tree and stochastic gradient boosting. The out of test accuracy has been estimated for all the models and random forest turned out to be the most accurate.

The random forest method has then been used to predict the quality of 20 instances of the exercise. !!! Insert result.

Introduction

All the analysis is based on the dataset provided here: http://groupware.les.inf.puc-rio.br/har. In the dataset the quality of execution of the exercise is represented by the variable **classe**. The remaining variables represent either an identifier of the participant and time of the experiment, or the data collected from the accelerometers positioned on the belt, arm, forearm and dumbell.

As the first step of the analysis, the libraries and the two datasets (for training and for testing) are loaded:

```r
library(dplyr)
library(ggplot2)
library(caret)
library(rpart)

pml.training <- read.csv("./pml-training.csv", header=TRUE, na.strings=c("NA",""))
pml.testing <- read.csv("./pml-testing.csv", header=TRUE, na.strings=c("NA",""))
```

Data preparation

A quick visual ispection of the training set reveals that a lot vo the observation contain a high number of *NA*. In order to quantify the amount of *NA*, the following procedure is performed:

```r
check.na <- function(x) mean(is.na(x)) * 100
sum(sapply(pml.training, check.na) > 95) / ncol(pml.training)
```

```
## [1] 0.625
```

As can be seen, 62.5% of the variables have a number of *NA* higher than 95% of the total length of the data frame. In this cases *NA* imputation would be highly problematic, therefore it was deemed necessary to remove this variables from the dataset since they would contribute very little information.

```r
# New data frame containing only the variables with less than 95% NA
pml.training.pruned <- pml.training[sapply(pml.training, check.na) < 95]
```

The dataset contains also variables like time stamps, user IDs, etc... These variables do not add anything to the model, therefore they are eliminated from the dataset.

```r
col_to_remove <- grepl("X|user_name|timestamp|window", colnames(pml.training.pruned))
pml.training.pruned <- pml.training.pruned[!col_to_remove]
dim(pml.training.pruned)
```

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

This process reduced the number of variables to 53. Following it was also checked that the remaing variables possessed nonzero variance.

```r
# Check for near zero variance
nearZeroVar(pml.training.pruned)
```

```
## integer(0)
```

The result shows that all variables seems to have enough variance to justify their inclusion in the model.

Modelling

As the first step in the modelling, the training data set is split into training and test.

```r
set.seed(1234)
# Separate in training and test set
in_train <- createDataPartition(pml.training.pruned$classe, p = 0.7, list = F)
pml.train <- pml.training.pruned[in_train,]
pml.test <- pml.training.pruned[-in_train,]
```

Following the calculations for the decision tree, gradient boosting and random forest model. All the models were run with 3-fold cross validation. The choise of 3-fold CV as opposed to higher fold CV was mostly dictated by the limitation of the machine used for the computations. Higher orders of folding lenghtned the computation time, specially for random forest.
The cross-validation method is set once for all the models:

```r
train_control <- trainControl(method="cv", number=3)
```

To assess the goodness of each model, a prediction on the test set has been performed and the relative accuracy and confusion matrix will be plotted.

Decision tree model

The decision tree model was computed using the package **rpart**.

```r
set.seed(125)
model_dt <- rpart(classe ~ ., data=pml.train, method="class")
predict_dt <- predict(model_dt, newdata=pml.test, type = "class")
confusionMatrix_dt <- confusionMatrix(predict_dt, pml.test$classe)
confusionMatrix_dt$overall[1] # Accuracy
```

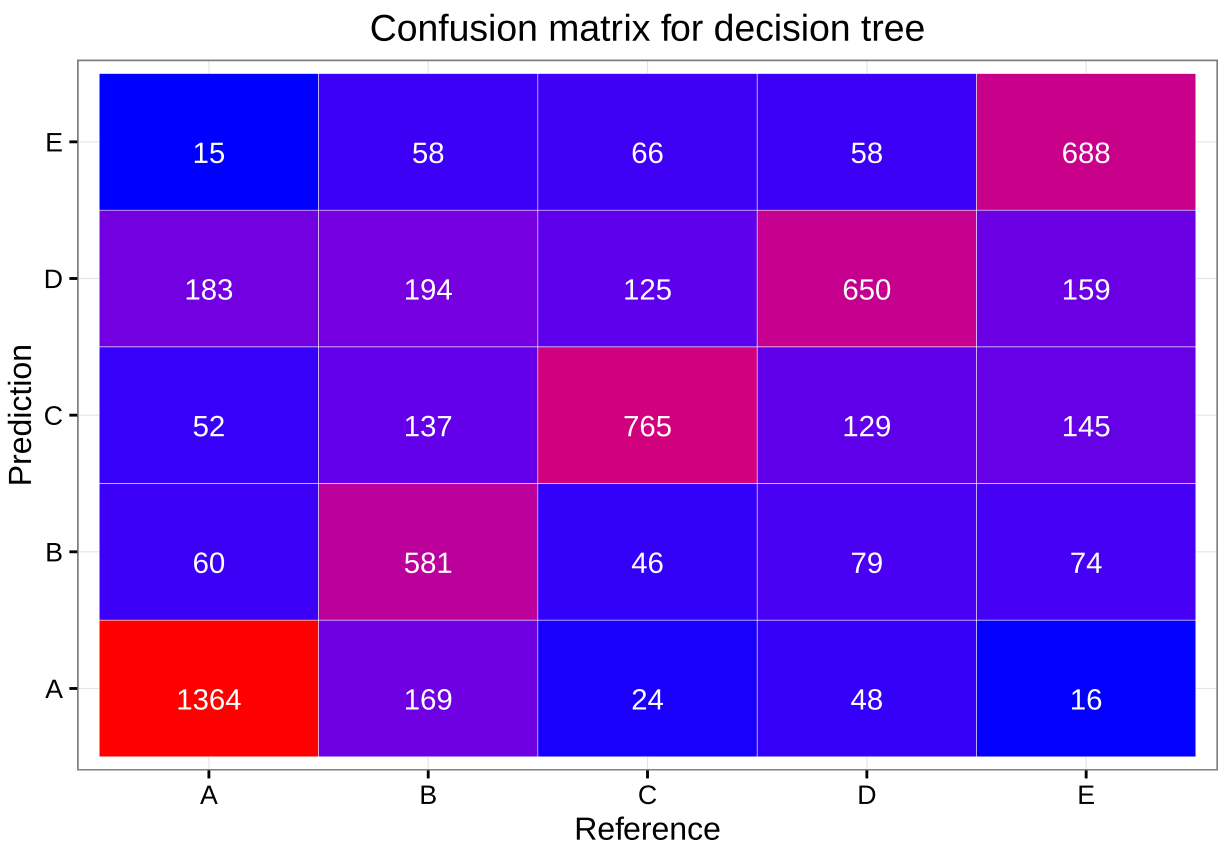
```
##  Accuracy
## 0.6878505
```

To plot the confusion matrix, a custom function was created:

```r
plot.matrix <- function(x,title) {
ggplot(as.data.frame(x$table), aes(x = Reference, y = Prediction)) +
        geom_tile(aes(fill = Freq), colour = "white") +
        geom_text(aes(label = sprintf("%1.0f", Freq)), colour = "white", vjust = 1) +
        scale_fill_gradient(low = "blue", high = "red") +
        theme_bw() + theme(legend.position = "none") +
                ggtitle(paste0("Confusion matrix for ", title))
}
```

The confusion matrix for the decision tree:

```r
plot.matrix(confusionMatrix_dt, "decision tree")
```

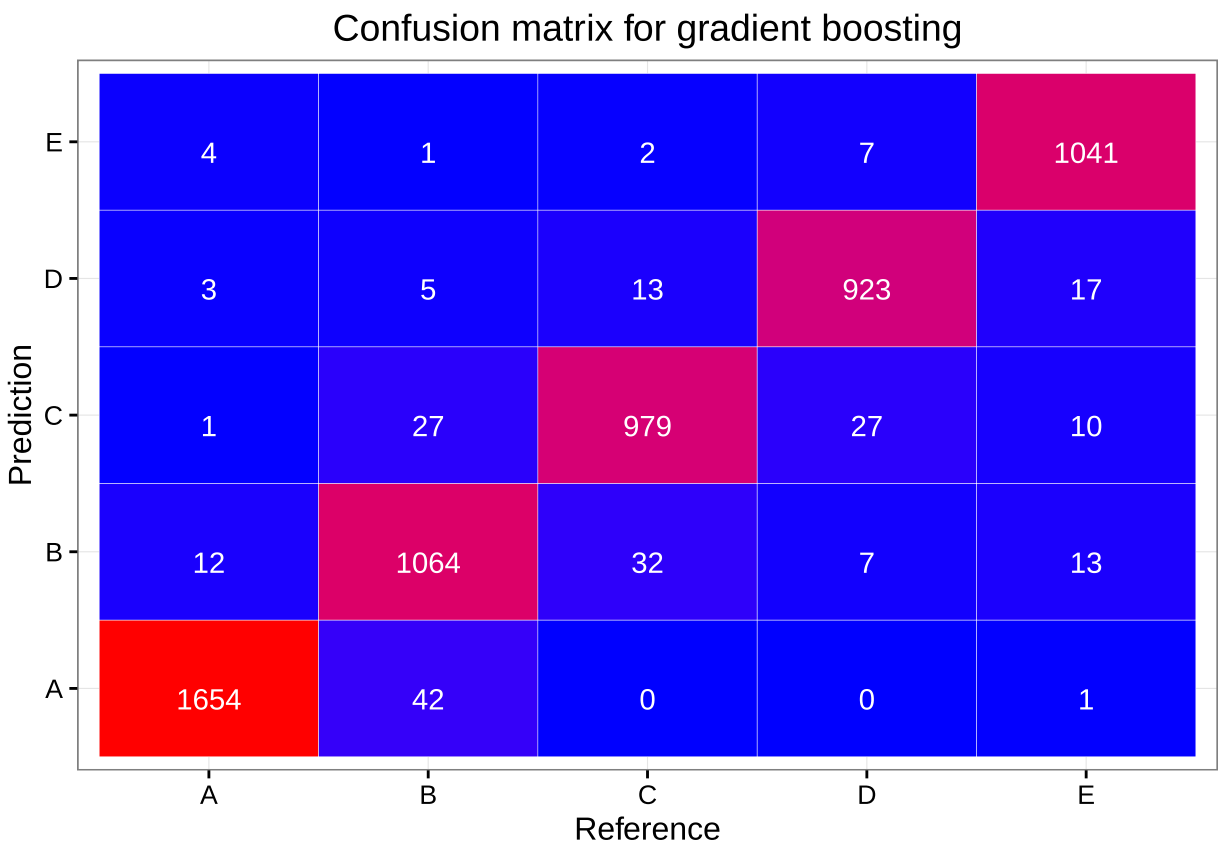


Gradient boosting

The same procedure is applied for the gradient boosting model, this time the **caret** package is used to train the model.

```
set.seed(123456)
model_gb <- train(classe~. , method="gbm", data=pml.train, trControl=train_control, verbose = FALSE)
predict_gb <- predict(model_gb, newdata=pml.test)
confusionMatrix_gb <- confusionMatrix(predict_gb, pml.test$classe)
```

```
##  Accuracy
## 0.9619371
```



Random forest

Finally, the model for random forest is computed.

```
set.seed(123)
model_rf <- train(classe~. , method="rf", data=pml.train, trControl=train_control)
predictRandForest <- predict(model_rf, newdata=pml.test)
confusionMatrix_rf <- confusionMatrix(predictRandForest, pml.test$classe)
```

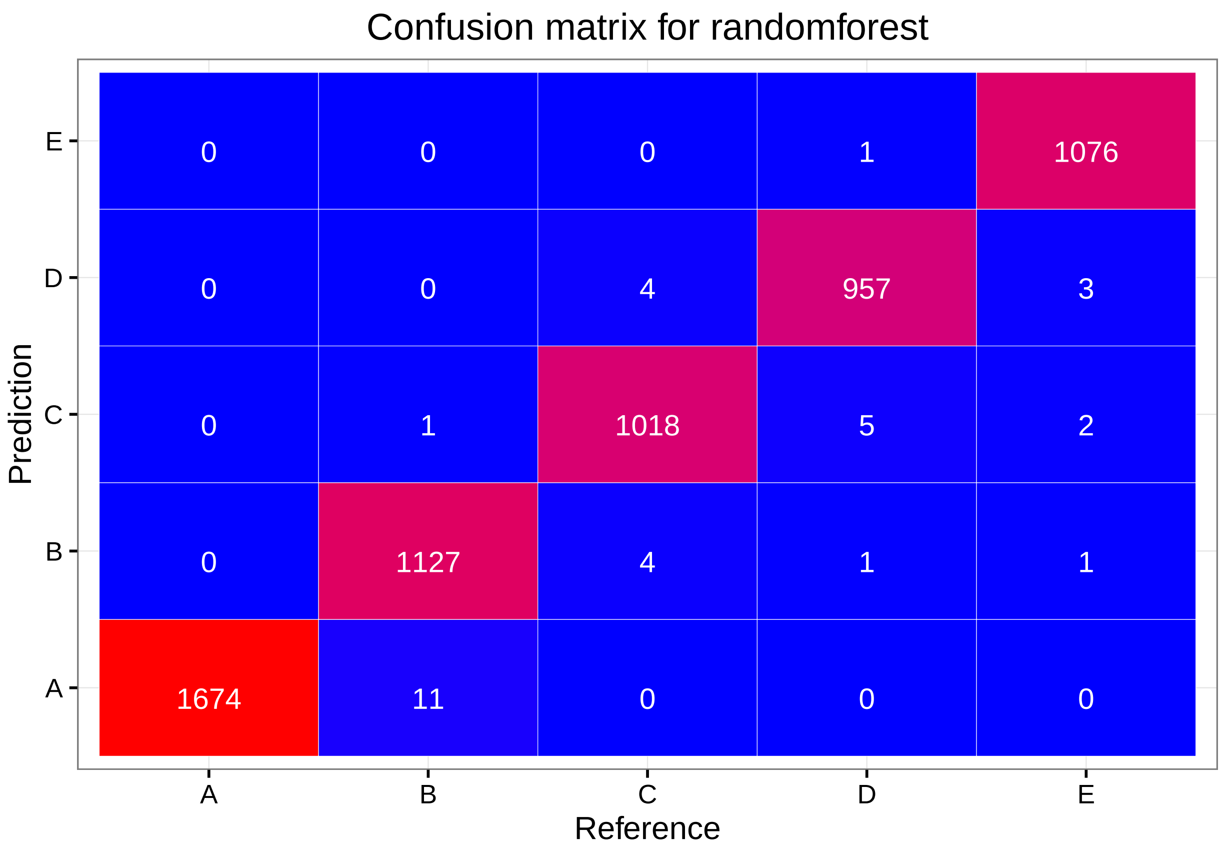
```
confusionMatrix_rf$overall[1] # Accuracy
```

```
##  Accuracy
## 0.9943925
```

```r
confusionMatrix_rf$overall[c(3,4)] # CI
```

```
## AccuracyLower AccuracyUpper
##      0.992134      0.996137
```

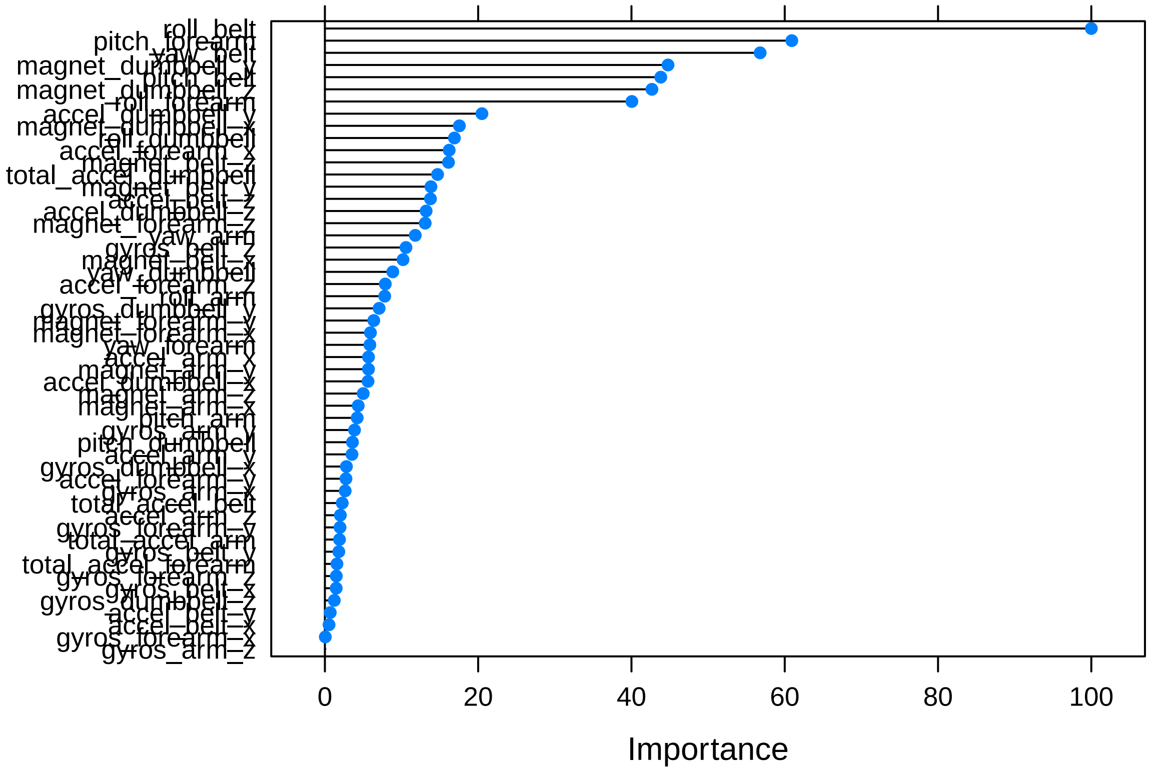
```r
plot.matrix(confusionMatrix_rf, "randomforest")
```



Conclusions

Comparing the 3 models we can see that the one providing the best out of test accuracy is random forest (model 3), with an accuracy of $99.4 \pm 0.2\%$. The expected error associated to this model is conversely: $0.6 \pm 0.2\%$. Following a plot of the variable importance:

```r
importance <- varImp(model_rf, scale=TRUE)
plot(importance)
```



Finally, the best model (random forest) is used to predict, the exercise quality for the 20 test observation:

```
predict_final_test <- predict(model_rf, newdata=pml.testing)
predict_final_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
```