Practical Machine Learning _ Prediction Assignment Writeup

Six participants were asked to do barbell lifts in five different ways, annotated through letters “A” to “E”. Thanks to devices such as Jawbone Up, Nike FuelBand and Fitbit, it is now possible to gather informations of their movements. The goal of this project is to see if it is possible to deduct the way in wich the barbell lift was done only thanks to detected patterns in these informations. This could be used to guide sportsmen in the way they exercise, to maximize the effect and avoid injuries.

Preparing data

```
library(caret)
```

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

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

```
library(parallel)
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)

test_data<-read.csv("pml-testing.csv")
train_data<-read.csv("pml-training.csv", na.strings=c("NA",""))
train_data <- train_data[sample(nrow(train_data)),-c(1,2)]

comp_var_test<-names(which(sapply(test_data, anyNA)))
comp_var_train<-names(which(sapply(train_data, anyNA)))
complete_train <- train_data[,!(names(train_data) %in% comp_var_train)]
testing<-test_data[,!(names(test_data) %in% comp_var_test)][,-c(1,2,60)]

intr<-createDataPartition(y=complete_train$classe,p=0.6, list=FALSE)
training<-complete_train[intr,]
valid<-complete_train[-intr,]
```

The dataset is composed of 19622 observations of 160 different variables. First of all, those variables are filtered to ignore those containing a majority of missing values, which would not help the diagnostic. Then, the data is split into two datasets : the training one on which we will train our machine learning algorithm, and the validation (valid) one on which we will test the accuracy of our trained model.
We also ignore some colums that will bring biaised information to the model, like the id number of the athlete, or his name. This gets the number of variables down to 58.

Pre-processing

```
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE)
```

As training times can sometimes take hours, we allow parallel training to speed it up. Moreover,we swich the original resampling method from bootstrapping to k-fold cross validation (“cv”). This way, the number of samples is reduced and the training time smaller.

Fit model

```
fit <- train(classe ~ ., data = training, method = "rf", trControl=fitControl)

stopCluster(cluster)
registerDoSEQ()
```

The training is done with the earlier described preprocessing and with the random forest algorithm, known for its robustness. It trains the “fit” model to predict the way the exercise was done (“classe” variable) when knowing the other 57 predictors.

```
fit
```

```
## Random Forest
##
## 11776 samples
##    57 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9421, 9422, 9420, 9421, 9420
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9898096  0.9871086
##   38    0.9981318  0.9976368
##   75    0.9968580  0.9960255
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 38.
```

As we can see, the trained random forest algorithm reaches an in-sample error of 99.8%, which is an excellent value. But the real accuracy that is of importance is the one on the valid dataset, that was not used for the training.

In-sample Error

```
confusionMatrix(predict(fit,training[,-58]),training[,58])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1926    0
##          E    0    0    0    4 2165
##
## Overall Statistics
##
##                Accuracy : 0.9997
##                  95% CI : (0.9991, 0.9999)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9996
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   0.9979   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   0.9996
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   0.9982
## Neg Pred Value         1.0000   1.0000   1.0000   0.9996   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1636   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1636   0.1842
## Balanced Accuracy      1.0000   1.0000   1.0000   0.9990   0.9998
```

If we evaluate the predictions of our model on the training data, we get an accuracy of 100%, which means that the model managed to find parameters perfectly assigning each sample to its class. That is a good thing, but an in-sample error of 100% does not mean anything if the model does not work on new data.

5-folds Cross-validation

```
fit$resample
```

```
##    Accuracy     Kappa Resample
## 1 0.9978769 0.9973143    Fold1
## 2 0.9995756 0.9994631    Fold3
## 3 0.9987256 0.9983881    Fold2
## 4 0.9974533 0.9967786    Fold5
## 5 0.9970276 0.9962399    Fold4
```

```
confusionMatrix.train(fit)
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 28.4  0.1  0.0  0.0  0.0
##          B  0.0 19.3  0.0  0.0  0.0
##          C  0.0  0.0 17.4  0.1  0.0
##          D  0.0  0.0  0.0 16.3  0.0
##          E  0.0  0.0  0.0  0.0 18.4
##
##  Accuracy (average) : 0.9981
```

The data is split into 5 “folds” to estimate how the model would perform in general on new data. The accuracy is each time excellent, giving us hope on the real out-sample error.
We can see on the cross-validated confusion matrix that no sample was incorrectly classified, giving an average accuracy of 99.8%, that we hope to see as high with new data.

Out of sample Error

```
confusionMatrix(predict(fit,valid[,-58]),valid[,58])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    0 1518    0    0    0
##          C    0    0 1368    0    0
##          D    0    0    0 1286    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9995, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

If we now try our model on the validation dataset untouched until now, we can see on the confusion matrix that there is now some errors on the predictions : 8 samples were incorrectly classified. 8/7846=0.001 however, so the out of sample error is really small compared to the number of samples. The resulting accuracy of our model is over 99.9%, which is excellent, almost perfect.