Practical Machine Learning: Course Project

Paul Byrne

12/08/2021

Executive Summary

The aim of the project is to use the data collected from personal activity monitors such as the Jawbone Up, Nike FuelBand, and Fitbit and specifically data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they conducted the excercise.

The subjects were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data Import and Initialisation

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

```
library(caret)
library(randomForest)
library(dplyr)

trainingURL <- url('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv')
testingURL <- url('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv' )

training_data <- read.csv(trainingURL)
testing_data <- read.csv(testingURL)
```

Data Cleansing and Selecting Variables

We next decide which variables to use in the prediction exercise. Initially those vairables which have a high correlation to each other, those with numerous missing entries and variables such as Name and ID which have no predictive value are removed.

Remove Columnms with Non Zero Variance

```
nonZeroVariance <- caret::nearZeroVar(training_data, allowParallel = TRUE)
training_data <- training_data[, -nonZeroVariance]
testing_data <- testing_data[, -nonZeroVariance]
```

Remove Columns with over 90% NAs

```
na_data <- which(colMeans(!is.na(training_data)) > 0.9)
training_data <- training_data[, na_data]
testing_data <- testing_data[, na_data]

dim(training_data)
```

```
## [1] 19622    59
```

```
dim(testing_data)
```

```
## [1] 20 59
```

Remove non-numeric Variables

```
training_data <- training_data[, 8:59]
testing_data <- testing_data[, 8:59]
```

The resulting dataset has 52 variables to be included within the model.

Model Selection and Approach

We will now use the training set in order to train and tune our model to predict the classe variable. This project will initially split the training set into a training and validation set.

The model selected will be the random forest, using parallel computing to make it more efficient and also then will consider using cross validation to tune the model. The random forest method was used over other methods due to its increased accuracy, howeer, the longer computational time has been considered through the use of parallel processing and cross validation as opposed to bootstrapping.

Training Data Partition

The training dataset is split into a training data (60% of observations) and validation dataset (40% of observations). This allows us to test the model on the validation set and predict the expected out of sample error rates.

```
library(caret)
intrain <- createDataPartition(training_data$classe, p=0.6, list=FALSE)
training_dataset <- training_data[intrain ,]
validation_dataset <- training_data[-intrain ,]
```

Running and Tuning the Model

Initially we are going to setup the ability to use parallel computing to make the running of the random forest model more efficient.

Note the approach was taken from Leonard Greski's article which can be found here: https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md

```
library(parallel)
library(doParallel)

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

Setting up the Random Forest with 5 Fold Cross Validation. This was chosen versus the default bootstrapping method as it significantly reduced the running time of the model.

Given the model was deemend to be efficient enough, no pre-processing such as PCA has been implemented.

```
fitControl <- trainControl(method = "cv", number = 5, allowParallel = TRUE)
fit <- train( classe~. , method="rf",data=training_dataset,trControl = fitControl)

stopCluster(cluster)
registerDoSEQ()
```

Model Output

The below shows the output and accuracy of the final fitted model on the training data set.

```
fit
```

```
## Random Forest
##
## 11776 samples
##    51 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9421, 9420, 9421, 9421, 9421
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9877719  0.9845304
##   26    0.9898095  0.9871087
##   51    0.9763922  0.9701317
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 26.
```

```
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  0.2  0.0  0.0  0.0
##          B  0.1 19.0  0.2  0.0  0.0
##          C  0.0  0.1 17.2  0.2  0.0
##          D  0.0  0.0  0.1 16.1  0.1
##          E  0.0  0.0  0.0  0.0 18.3
##
##  Accuracy (average) : 0.9898
```

The model using random forest on the training set has an accuracy of 98.9%, meaning the in sample error is 1.1%

Validation Set

Next we use the model to predict on the validation set to test the out of sample accuracy of the model.

```
rf_prediction <- predict(fit, validation_dataset)
confusionMatrix(rf_prediction, validation_dataset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   12    0    0    0
##          B    1 1499    8    0    1
##          C    0    7 1356   11    0
##          D    0    0    4 1275    7
##          E    0    0    0    0 1434
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9915, 0.9952)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9918
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9875   0.9912   0.9914   0.9945
## Specificity            0.9979   0.9984   0.9972   0.9983   1.0000
## Pos Pred Value         0.9947   0.9934   0.9869   0.9914   1.0000
## Neg Pred Value         0.9998   0.9970   0.9981   0.9983   0.9988
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
```

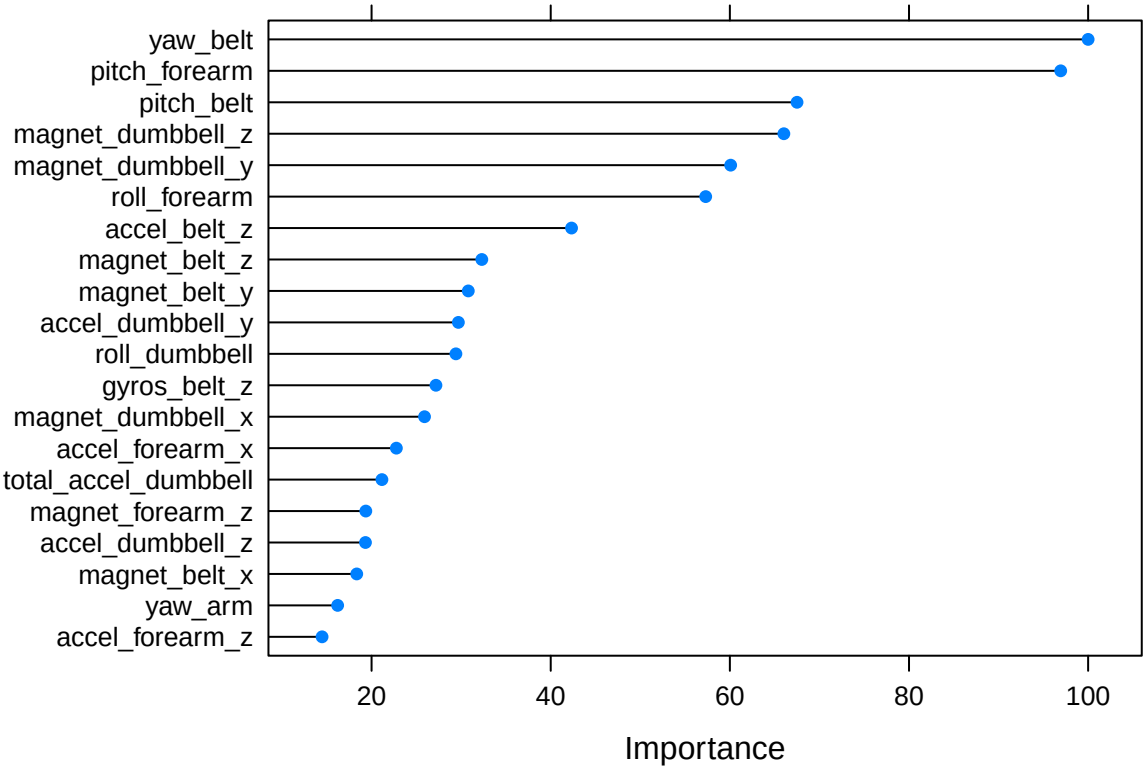
```
## Detection Rate         0.2843   0.1911   0.1728   0.1625   0.1828
## Detection Prevalence   0.2859   0.1923   0.1751   0.1639   0.1828
## Balanced Accuracy      0.9987   0.9930   0.9942   0.9949   0.9972
```

Accuracy of the model using the validation dataset was 99.1%, therefore the out of sample error is 0.9%.

Variable Importance

Next we consider the vairable importance. We can see from the below plot that the yaw_belt and pitch_forarm and the two most important variables in determining the type of exercise being undertaken from the model.

```
plot(varImp(fit), top = 20)
```



Predicting on the Test Set

The model constructed above has then been use to predict using the 20 observations in the testing set. This code and the results have been hidden for the final course project.

Conclusion

We have constructed a random forest model, using 5 fold cross validation to predict the type of exercise completed, given the 51 variables given in the dataset as measured by the fitness trackers.

The in sample and out of sample errors were both c.99% using the training and validation datasets. When viewing the variable importance it is apparent that the Yaw Belt and Forearm Pitch were the most important in determining the type of activity.