Human Activity Recognition Prediction assignment

R Mofidi

30/06/2020

Introduction

In the last decade Human Activity Recognition (HAR) - has emerged as a key research area in wearable device technology. In particular for the development of context-aware systems. There are many potential applications for HAR, These include life log systems for monitoring energy expenditure and supporting weight-loss programs, and digital assistants for weight lifting amongst others. They promote healthy living and weight loss programs.

Using devices such as Nike FuelB (TM), Apple (TM) iWatch and Fitbit(TM), it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are technology enthusiasts. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

In this project, our aim was to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. Detailed information about how the data is collected is available from the following website: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

The goal of this project was to predict the manner in which they did the exercise. This is recorded in the variable *classe* in the training set. This report describe how the model was build trained and cross validated,

The prediction model is used to predict what activity was carried out 20 different test cases.

Methods

Loading and Opening the training and out of sample testing datasets

The first step of the data analysis task involves loading and opening the appropriate packages and libraries used to develop the classification models. The powerful and versatile "caret" machine learning package was employed for this assignment, The "Rattle" package was used to illustrate the classification tree " and "randomForest" package was used to develop the random forest classifier.

```r
library (lattice)
library(ggplot2)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2
```

```r
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 4.0.2
```

```
## 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.
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.2
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
##
##     importance
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(e1071)
```

Downloading the data sets

The next step involved downloading the training and testing data sets:

```r
if(!file.exists("~/data")){dir.create("~/data")}
fileUrltr <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(fileUrltr,destfile="~/data/pml-training")
fileUrltest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(fileUrltest,destfile="~/data/pml-testing")
trainingAS<- read.csv("~/data/pml-training", sep=",")
testingAS<- read.csv("~/data/pml-testing", sep=",")
```

Examining and visualizing the data

The testing and training data sets include 160 different variables with many variables containing NA entries or 0 entries. There are 5 different outcome variables which are listed under the variable classe (variable-160). The variable "classe" was a character variable which is made up of letters A, B, C, D and E.

Prior to attempting to design the predictive models, the data sets needed to be cleaned. in the first instance this involved removing variables which contain mostly NA entries or have near zero variance:

```
zeroVar<-nearZeroVar(trainingAS)
trainingAS<- trainingAS[,-zeroVar]
testingAS<- testingAS[,-zeroVar]

dim(trainingAS); dim(testingAS)
```

Removing variables with Near zero variance (non-classifiers)

```
## [1] 19622   100
```

```
## [1]  20 100
```

```
MNA<- sapply(trainingAS, function(x) mean(is.na(x)))>0.90
trainingAS<- trainingAS[,MNA==FALSE]
testingAS<- testingAS[,MNA==FALSE]

dim(trainingAS); dim(testingAS)
```

Remove All NA variables

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

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

Removing variables which do not contribute to data analysis: Variables 1 to 6 are for case identification and are unlikely to contribute to the predictive ability of the machine learning models. They include:

1- number 2-user_name 3-raw_timestamp_part_1
4-raw_timestamp_part_2
5- cvtd_timestamp 6- new_window

They were therefore excluded from the training and testing datasets:

```
trainingAS<- trainingAS[,-c(1:6)]
testingAS<- testingAS[,-c(1:6)]
dim(trainingAS); dim(testingAS)
```

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

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

```
trainingAS1<- trainingAS
trainingAS1$classe<- as.factor(trainingAS1$classe)
trainingAS1$classe<- as.integer(trainingAS1$classe)
```

Changing the classe variable from character to integer

Assess and Visualize the relationships

In order to examine the relationships between the variable "classe" and the remaining 53 variables multivariate regression analysis was performed so that variables which bear no correlation with the "classe" variable are identified and removed if necessary:

```
summary(lm(formula = classe ~ ., data = trainingAS1))
```

```
##
## Call:
## lm(formula = classe ~ ., data = trainingAS1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4218 -0.6928 -0.0619  0.6462  5.8030
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             8.264e+00  4.292e-01  19.252  < 2e-16 ***
## roll_belt               2.596e-02  2.004e-03  12.955  < 2e-16 ***
## pitch_belt              3.820e-02  1.930e-03  19.789  < 2e-16 ***
## yaw_belt               -3.692e-03  3.994e-04  -9.243  < 2e-16 ***
## total_accel_belt        5.120e-02  6.702e-03   7.639 2.29e-14 ***
## gyros_belt_x            2.110e-01  7.223e-02   2.921 0.003490 **
## gyros_belt_y           -5.847e-01  1.932e-01  -3.027 0.002475 **
## gyros_belt_z            1.999e-01  5.152e-02   3.880 0.000105 ***
## accel_belt_x           -6.168e-03  1.183e-03  -5.212 1.88e-07 ***
## accel_belt_y           -2.576e-02  1.161e-03 -22.197  < 2e-16 ***
## accel_belt_z            2.515e-03  1.112e-03   2.262 0.023730 *
## magnet_belt_x          -2.538e-03  4.507e-04  -5.631 1.82e-08 ***
## magnet_belt_y          -1.392e-02  5.815e-04 -23.938  < 2e-16 ***
## magnet_belt_z           2.415e-03  2.995e-04   8.061 7.98e-16 ***
## roll_arm               -4.640e-05  1.474e-04  -0.315 0.752937
## pitch_arm              -4.240e-03  3.386e-04 -12.520  < 2e-16 ***
## yaw_arm                 5.263e-04  1.228e-04   4.284 1.84e-05 ***
## total_accel_arm         1.179e-02  1.088e-03  10.836  < 2e-16 ***
## gyros_arm_x             1.050e-01  1.133e-02   9.268  < 2e-16 ***
## gyros_arm_y             1.096e-01  2.764e-02   3.966 7.34e-05 ***
## gyros_arm_z            -7.671e-02  2.127e-02  -3.606 0.000311 ***
## accel_arm_x             5.756e-04  2.226e-04   2.585 0.009737 **
## accel_arm_y            -3.947e-03  4.940e-04  -7.990 1.42e-15 ***
## accel_arm_z             6.427e-03  2.815e-04  22.828  < 2e-16 ***
## magnet_arm_x           -4.097e-04  7.808e-05  -5.246 1.57e-07 ***
## magnet_arm_y           -1.405e-03  2.003e-04  -7.015 2.38e-12 ***
## magnet_arm_z         -8.539e-04  1.134e-04  -7.532 5.20e-14 ***
## roll_dumbbell         2.862e-03  2.056e-04  13.919  < 2e-16 ***
## pitch_dumbbell       -2.332e-03  4.419e-04  -5.277 1.33e-07 ***
## yaw_dumbbell         -5.178e-03  2.130e-04 -24.306  < 2e-16 ***
## total_accel_dumbbell  3.855e-02  2.155e-03  17.882  < 2e-16 ***
## gyros_dumbbell_x      2.025e-01  2.998e-02   6.754 1.48e-11 ***
## gyros_dumbbell_y      1.750e-01  1.871e-02   9.352  < 2e-16 ***
## gyros_dumbbell_z      6.879e-02  2.055e-02   3.347 0.000819 ***
## accel_dumbbell_x      7.156e-03  3.942e-04  18.153  < 2e-16 ***
## accel_dumbbell_y      3.247e-04  3.542e-04   0.917 0.359379
## accel_dumbbell_z      8.819e-04  2.149e-04   4.103 4.09e-05 ***
## magnet_dumbbell_x    -3.566e-03  1.041e-04 -34.254  < 2e-16 ***
## magnet_dumbbell_y    -8.364e-04  8.712e-05  -9.601  < 2e-16 ***
## magnet_dumbbell_z     1.037e-02  1.579e-04  65.683  < 2e-16 ***
## roll_forearm          8.124e-04  8.764e-05   9.270  < 2e-16 ***
## pitch_forearm         1.132e-02  4.543e-04  24.922  < 2e-16 ***
## yaw_forearm          -3.505e-04  9.593e-05  -3.654 0.000259 ***
## total_accel_forearm   2.263e-02  9.875e-04  22.913  < 2e-16 ***
## gyros_forearm_x      -5.405e-02  2.051e-02  -2.636 0.008401 **
## gyros_forearm_y      -4.480e-03  6.297e-03  -0.712 0.476751
## gyros_forearm_z       3.722e-02  1.859e-02   2.001 0.045361 *
## accel_forearm_x       9.356e-04  1.634e-04   5.726 1.05e-08 ***
## accel_forearm_y       8.205e-04  1.138e-04   7.211 5.77e-13 ***
## accel_forearm_z      -6.091e-03  1.267e-04 -48.088  < 2e-16 ***
## magnet_forearm_x     -7.436e-04  6.906e-05 -10.767  < 2e-16 ***
## magnet_forearm_y     -5.330e-04  4.595e-05 -11.600  < 2e-16 ***
## magnet_forearm_z      1.697e-04  4.347e-05   3.905 9.45e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.017 on 19569 degrees of freedom
## Multiple R-squared:  0.5264, Adjusted R-squared:  0.5251
## F-statistic: 418.2 on 52 and 19569 DF,  p-value: < 2.2e-16
```

Multivariate regression analysis failed to reveal a significant correlation between classe and following variables: 1- roll_arm
2-accel_dumbbell_y 3- gyros_forearm_y

The relationship between each of these 3 variables and classe variable was examined using univariate regression analysis to see if there is any correlation between these 3 variables and the variable "classe"

```r
summary(lm(trainingAS1$classe~trainingAS1$roll_arm))
```

```
##
## Call:
## lm(formula = trainingAS1$classe ~ trainingAS1$roll_arm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.05775 -1.50633  0.09842  1.26243  2.57727
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.7375738  0.0108036  253.39   <2e-16 ***
## trainingAS1$roll_arm 0.0017788  0.0001442   12.33   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.47 on 19620 degrees of freedom
## Multiple R-squared:  0.007691,   Adjusted R-squared:  0.00764
## F-statistic: 152.1 on 1 and 19620 DF,  p-value: < 2.2e-16
```

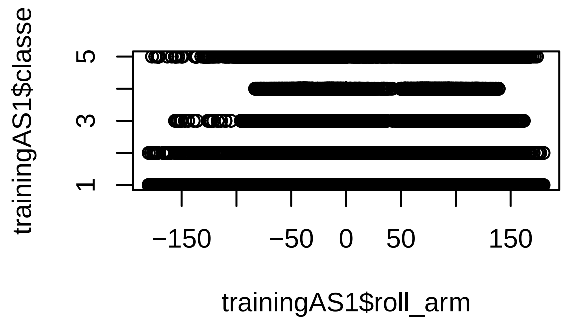
```r
summary(lm(trainingAS1$classe~trainingAS1$accel_dumbbell_y))
```

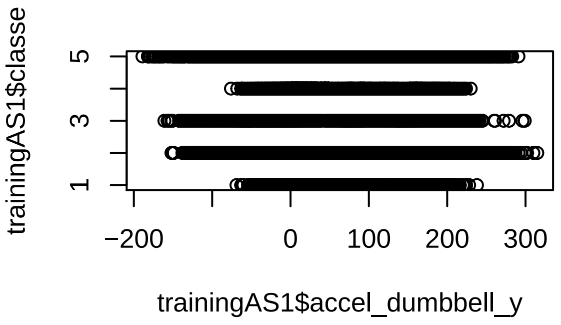
```
##
## Call:
## lm(formula = trainingAS1$classe ~ trainingAS1$accel_dumbbell_y)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8045 -1.7405  0.1992  1.2288  2.2998
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   2.7845397  0.0125721 221.485   <2e-16 ***
## trainingAS1$accel_dumbbell_y -0.0002897  0.0001304  -2.221   0.0263 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.475 on 19620 degrees of freedom
## Multiple R-squared:  0.0002514,   Adjusted R-squared:  0.0002005
## F-statistic: 4.935 on 1 and 19620 DF,  p-value: 0.02634
```

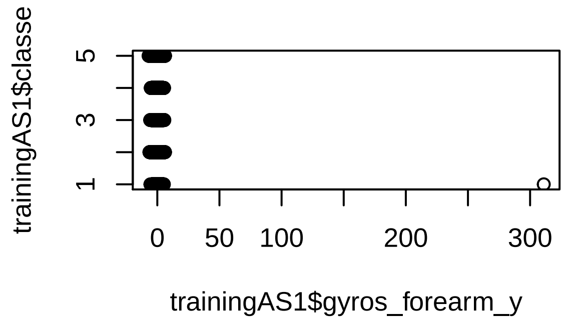
```r
summary(lm(trainingAS1$classe~trainingAS1$gyros_forearm_y))
```

```
##
## Call:
## lm(formula = trainingAS1$classe ~ trainingAS1$gyros_forearm_y)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7963 -1.7599  0.2184  1.2334  2.2597
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  2.769650   0.010536 262.871   <2e-16 ***
## trainingAS1$gyros_forearm_y -0.004796   0.003397  -1.412    0.158
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.475 on 19620 degrees of freedom
## Multiple R-squared:  0.0001016,   Adjusted R-squared:  5.061e-05
## F-statistic: 1.993 on 1 and 19620 DF,  p-value: 0.158
```

It turns out that these 3 variables are excellent classifiers of the classe variable on their own. Therefore there are kept for training of the eventual model.



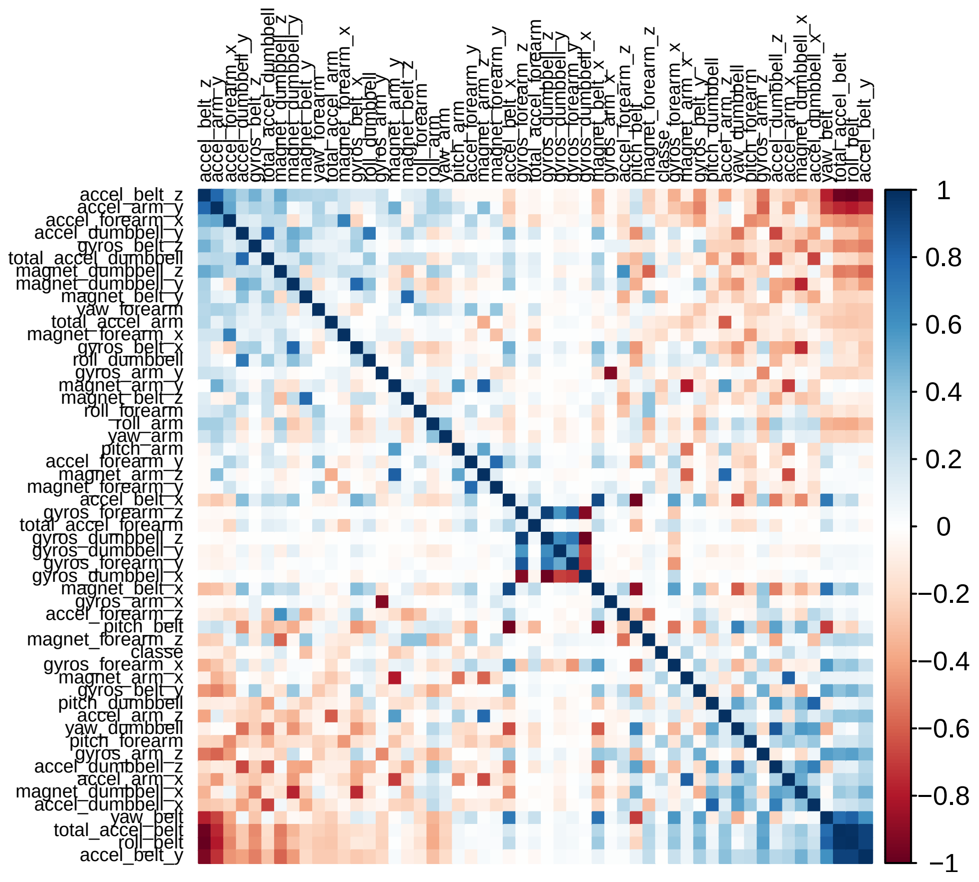




Visualizing the relationship between the input variables

The following plot is a visual representation of the relationships between the covariates. The darker the colour the stronger the relationships. Blue colour denotes a positive relationship whilst red/brown colour denotes negative relationships.

```
## corrplot 0.84 loaded
```



The optimum Machine learning Model

In order to develop the optimum classification model for performing the predictions associated with this task we used the following models:

1- A random forest classifier

2- A Classification and regression tree

If one of these models proved adequate no further analysis would be performed if not, other machine learning classifiers would be tried. Each model was trained using a proportion (70%) of the training data and validated using a cross Validation dataset which ws 30% of the training dataset. The final model was then tested on testingAS to assess the generalisability of each classifier. This may be seen as a superfluous step but it is the only means of cross validation.The training dataset is called "trainingAS-t". The cross validation dataset is called "crossVal"

```
trainingAS$classe<- as.factor(trainingAS$classe)
inTrain<- createDataPartition(y=trainingAS$classe, p=0.7, list =FALSE)
trainingAS_t<- trainingAS[inTrain,]
crossVal<- trainingAS[-inTrain,]
```

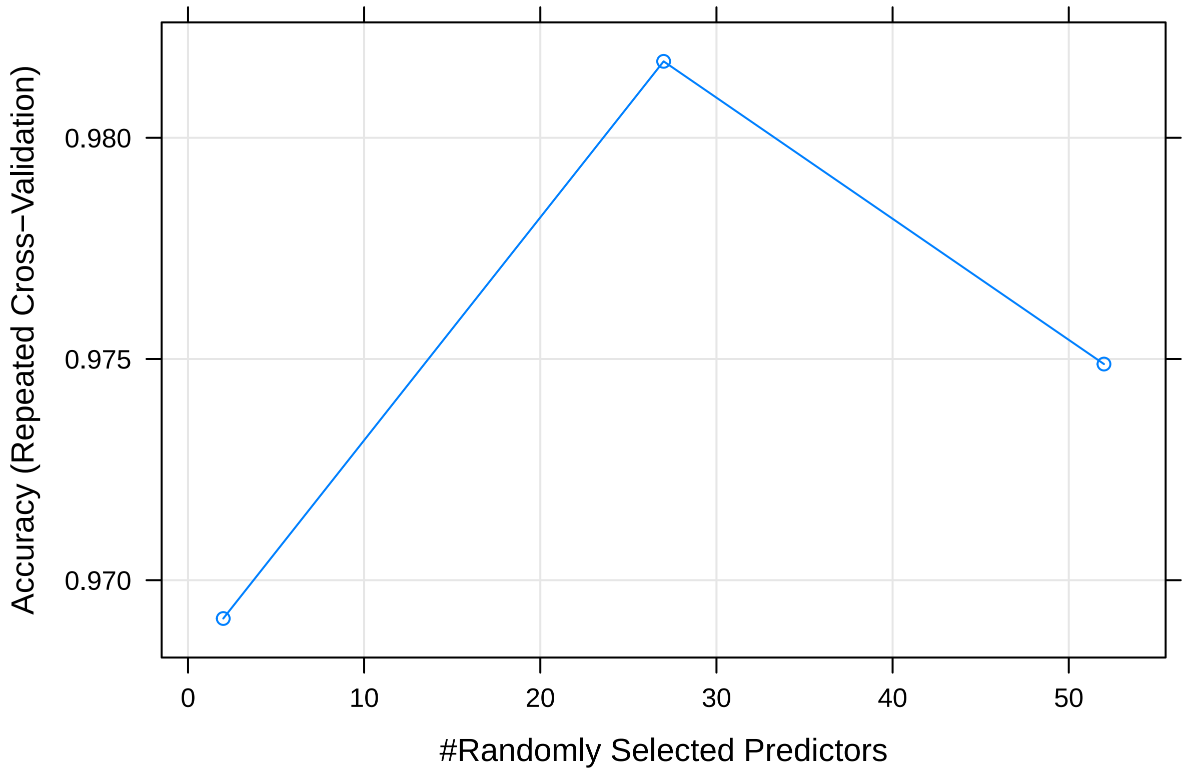
Random Forest Classifier

The first model which we develop was a random forest classifier. The random forest classifier performs repeated internal sampling and cross validation, however in order to maintain a level playing field between the classifiers it was trained and cross validated using the data partitions used:

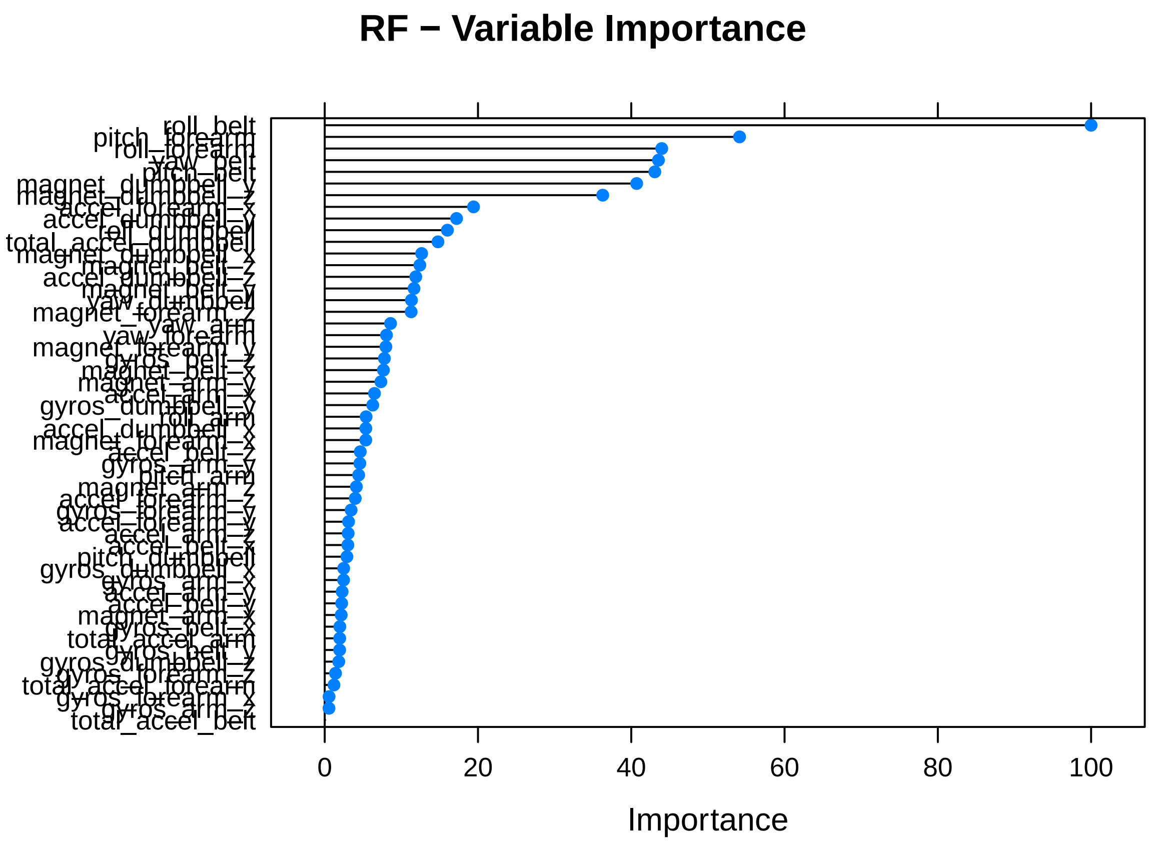
```r
set.seed(301)
tr <- trainControl(method="repeatedcv", number=3, verboseIter=FALSE)
modFitRF <- train(classe ~ ., data=trainingAS_t, method="rf",
                          trControl=tr, ntree=10)
modFitRF
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 1 times)
## Summary of sample sizes: 9159, 9157, 9158
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9691339  0.9609499
##   27    0.9817279  0.9768841
##   52    0.9748856  0.9682248
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```r
plot(modFitRF)
```



It was possible to examine the relative importance of each variable in constructing the eventual random forest classifier. The following is a graphical representation of this:



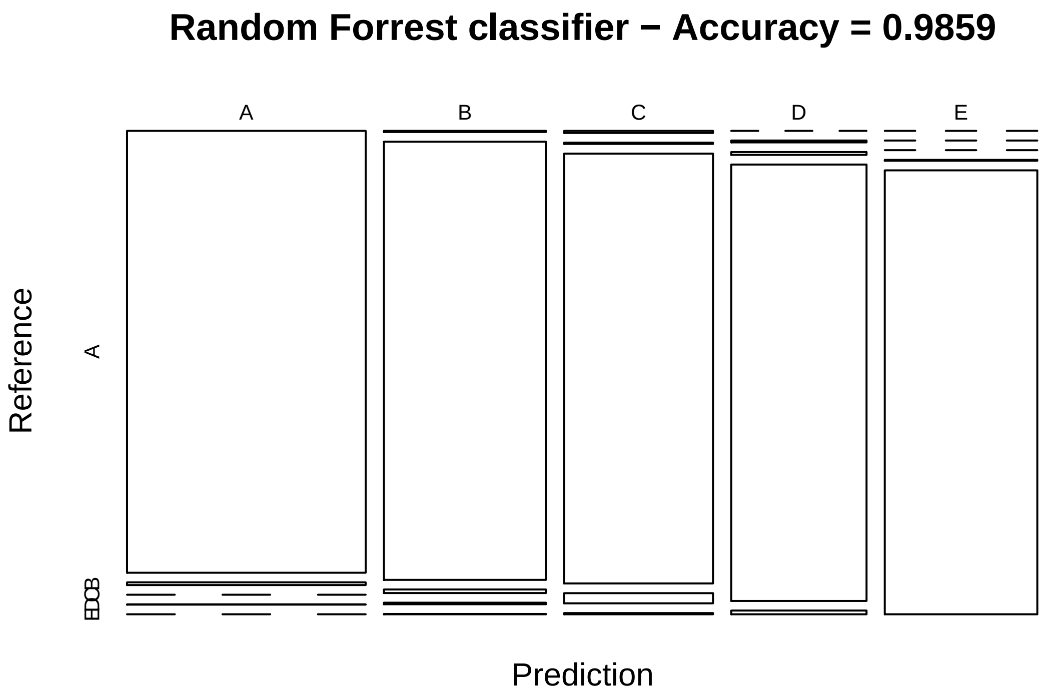
Assessing the accuracy of the random forest classifier using the cross validation dataset

```
predCV_rf<- predict(modFitRF, newdata = crossVal, type="raw")
MatrixRF<- confusionMatrix(predCV_rf, crossVal$classe)
MatrixRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1666   10    0    1    0
##          B    3 1122    9    4    1
##          C    5    3 1011   24    3
##          D    0    4    6  933    8
##          E    0    0    0    2 1070
##
## Overall Statistics
##
##                Accuracy : 0.9859
##                  95% CI : (0.9825, 0.9888)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9822
```

```
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9952   0.9851   0.9854   0.9678   0.9889
## Specificity            0.9974   0.9964   0.9928   0.9963   0.9996
## Pos Pred Value         0.9934   0.9851   0.9665   0.9811   0.9981
## Neg Pred Value         0.9981   0.9964   0.9969   0.9937   0.9975
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2831   0.1907   0.1718   0.1585   0.1818
## Detection Prevalence   0.2850   0.1935   0.1777   0.1616   0.1822
## Balanced Accuracy      0.9963   0.9907   0.9891   0.9821   0.9942
```

The following graph is a visual representation of the classification matrix for the random forest classifier and illustrates the accuracy of the random forest classifier at correctly classifying the cross validation sample into their correct "classe".



Finally we perform the predictions using the trained and cross validated model:

```
predRandomForest<- predict(modFitRF, newdata = testingAS, type="raw")
predRandomForest
```

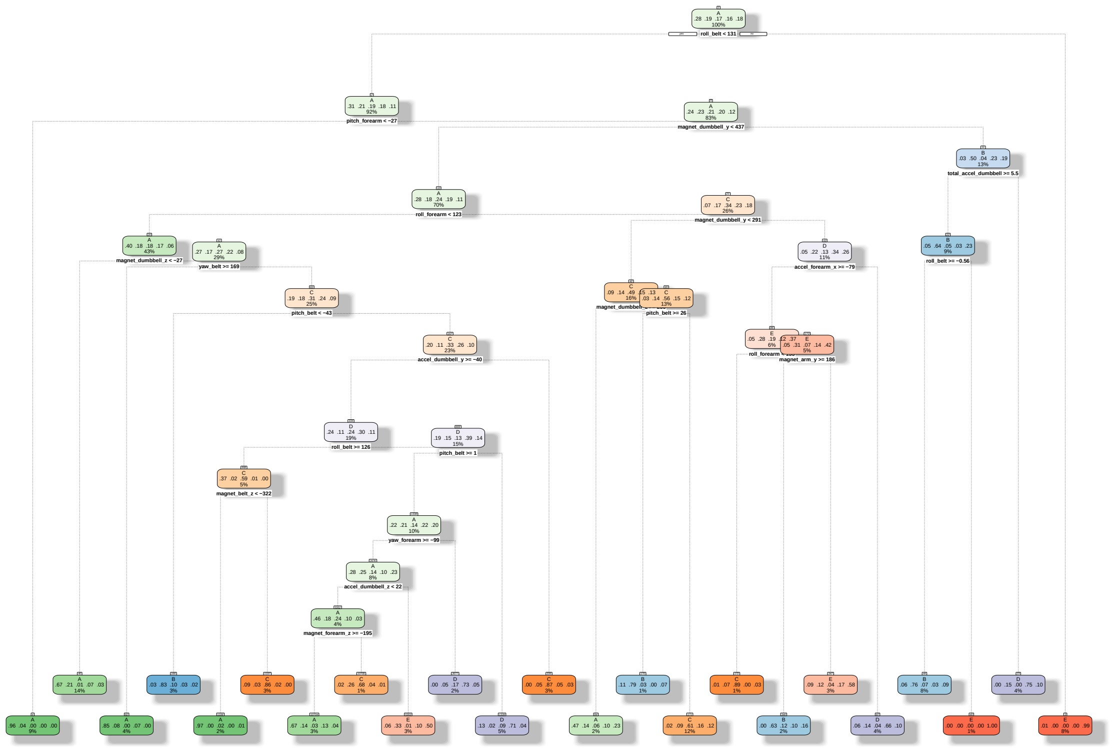
```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

Classification Tree (Decision Tree) classifier

For comparison purposes a classification tree was also created and trained using the same partitioning of the dataset to perform the exact same prediction. The following graph illustrates the structure of the decision tree classifier:

```r
set.seed(333)
library(rpart)
library(rpart.plot)
trainingAS_t$classe<- as.factor(trainingAS_t$classe)
modfitDTree<- rpart(classe~., data=trainingAS_t, method="class")
fancyRpartPlot(modfitDTree)
```

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



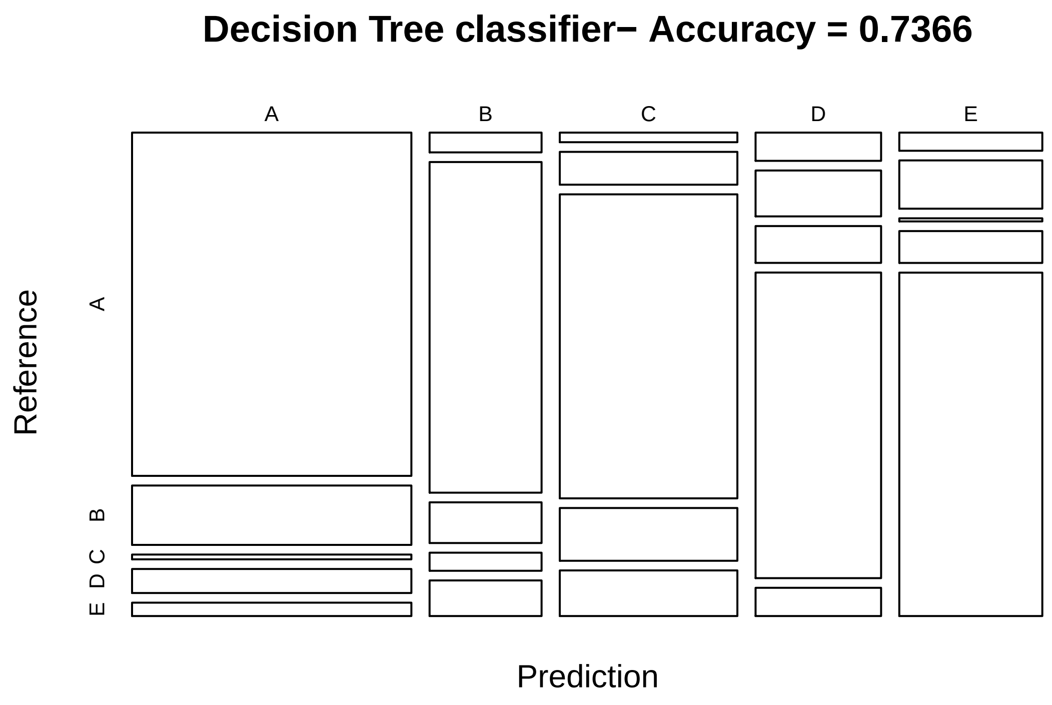
Cross Validation of the Decision Tree Classifier The accuracy of prediction using the tree classifier was assessed using the cross validation dataset. A confusion matrix was created for this purpose:

```r
predTreeCV<- predict(modfitDTree, newdata = crossVal, type="class")
mtr<- confusionMatrix(predTreeCV, crossVal$classe)
mtr
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1515  262   21  106   59
##          B   35  585   72   32   63
##          C   27   92  853  148  128
##          D   56   91   73  606   56
##          E   41  109    7   72  776
##
## Overall Statistics
##
##                Accuracy : 0.7366
##                  95% CI : (0.7252, 0.7478)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.665
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9050  0.51361   0.8314   0.6286   0.7172
## Specificity            0.8936  0.95744   0.9187   0.9439   0.9523
## Pos Pred Value         0.7718  0.74333   0.6835   0.6871   0.7721
## Neg Pred Value         0.9595  0.89133   0.9627   0.9284   0.9373
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2574  0.09941   0.1449   0.1030   0.1319
## Detection Prevalence   0.3336  0.13373   0.2121   0.1499   0.1708
## Balanced Accuracy      0.8993  0.73552   0.8750   0.7863   0.8348
```

The following graph is a visual representation of the classification matrix for the decision tree classifier and illustrates the accuracy of the decision tree classifier at correctly classifying the cross validation sample into their correct “classe”.



Finally a similar prediction i.e. the prediction of the "classe" variable in the testingAS data set was performed using the decision tree classifier:

```
predTree<- predict(modfitDTree, newdata = testingAS, type="class")
predTree
```

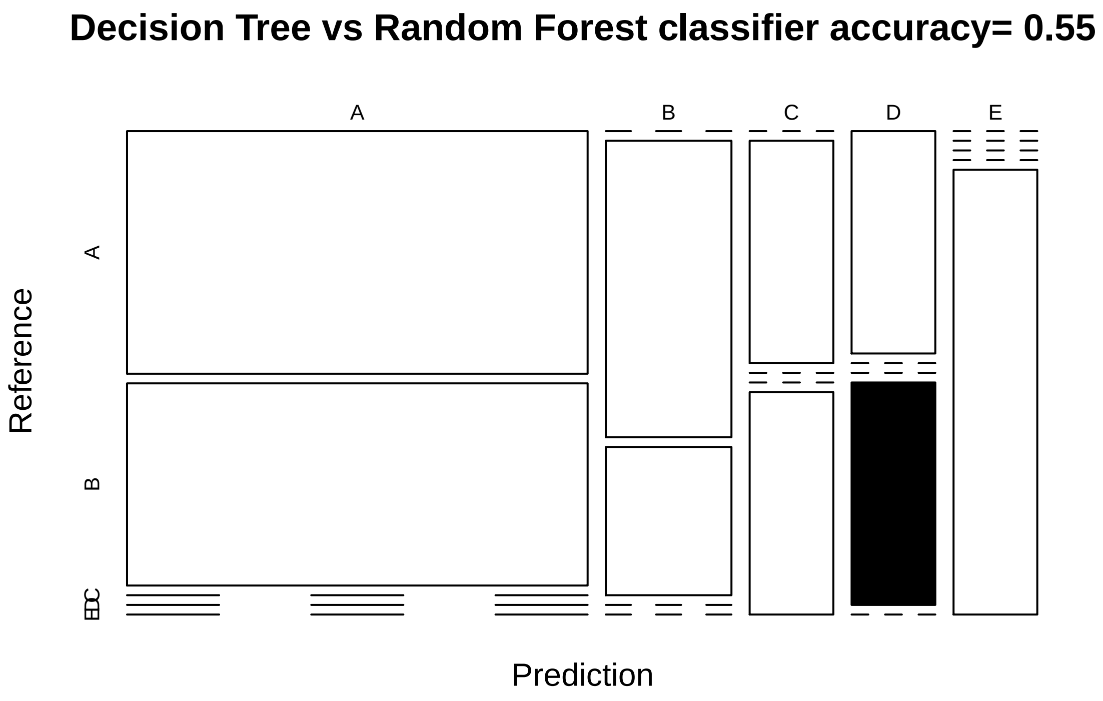
```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  A  D  A  C  D  A  A  A  C  B  A  A  E  E  A  A  A  B
## Levels: A B C D E
```

The accuracy of Decision tree classifier at predicting the "class" variable was compared with the random forest model acting as gold standard.

```
mtr2<- confusionMatrix(predTree, predRandomForest)
mtr2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction A B C D E
##          A 6 5 0 0 0
##          B 0 2 1 0 0
##          C 0 1 0 0 1
##          D 1 0 0 1 0
##          E 0 0 0 0 2
```

```
##
## Overall Statistics
##
##                Accuracy : 0.55
##                  95% CI : (0.3153, 0.7694)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : 0.1275
##
##                   Kappa : 0.3772
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8571   0.2500   0.0000   1.0000   0.6667
## Specificity            0.6154   0.9167   0.8947   0.9474   1.0000
## Pos Pred Value         0.5455   0.6667   0.0000   0.5000   1.0000
## Neg Pred Value         0.8889   0.6471   0.9444   1.0000   0.9444
## Prevalence             0.3500   0.4000   0.0500   0.0500   0.1500
## Detection Rate         0.3000   0.1000   0.0000   0.0500   0.1000
## Detection Prevalence   0.5500   0.1500   0.1000   0.1000   0.1000
## Balanced Accuracy      0.7363   0.5833   0.4474   0.9737   0.8333
```



Conclusions

As one would expect the random forest classifier was significantly better than the classification tree at correctly classifying the "classe" variable. This is not surprising as ensemble classifiers such as random forests and adaptive boosting algorithms are some of the most accurate classifiers in use today, both for classification and regression problems. They achieve this level of accuracy by using the collective power of imperfect "weak" classifiers (3).It is acknowledged that decision trees are not as powerful as random forest classifiers however the comparison between the two is still a useful exercise.

Random forests belong to a group of machine learning algorithms called ensemble classifiers. They work through cooperation of many 'weak classifiers' to improve the predictive ability of the final classifier. Random forests work through a process called bagging where each member of ensemble simultaneously provides a contribution towards the final predictive model, whilst the other major class of ensemble classifiers or adaptive boosting algorithms improve the accuracy of classification through iterative improvements in the classification process. The reason why I selected random forest algorithm rather than an adaboost (adaptive boosting classifier) is that the latter are prone to noise and sensitive to outlying variables and therefore need to be used with caution. In this task the random forest classifier was 100% correct at classifying the 20 test samples which were part of the quiz.

References:

1- W Ugulino, E Velloso, H Fuks. Human Activity Recognition dataset. http://groupware.les.inf.puc-rio.br/har

2-Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6.

3- Breiman L, Cutler A. Random Forest Classifiers. https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm