Machine Learning Project Assignment

Weight lifting exercise - Predict if exercises are done correctly

Background Using devices such as Jawbone Up, Nike FuelBand, and Fitbit 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 tech geeks. 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, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Reference 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. http://groupware.les.inf.puc-rio.br/work.jsf?p1=10335 (http://groupware.les.inf.puc-rio.br/work.jsf?p1=10335)

Objective Develop a prediction model using the training data set to predict Classe.

Summary A prediction model was developed based on random forest which did very well in cross validation and was able to classify all testing observations (20) correctly

Loading & Exploring Data

```
# Loading libraries & enable parallel processing
library(caret);library(gbm);library(randomForest);library(doParallel)
set.seed(12345)
cl <- makeCluster(detectCores());registerDoParallel(cl)
```

```
training.rd <- read.csv("C:/Users/Werner/Documents/rcourse/pml01/pmldata/pml-training.csv")
testing.rd <- read.csv("C:/Users/Werner/Documents/rcourse/pml01/pmldata/pml-testing.csv")
```

```
dim(training.rd);dim(testing.rd);
```

```
## [1] 19622   160
```

```
## [1]  20 160
```

With 19622 rows and 160 columns the training data set is huge. Using commands like View(training.rd) and edit(training.rd) the need for cleaning data is obvious (empty columns, NA columns). Also columns 1 to 7 are not related to measurements. Not related columns and NA columns are eliminated. In case I have trouble to develop a very predictive model I would come back to the data cleaning activities. As the number of columns is huge I also use nearZeroVar(t) to eliminate columns with small variances (no or small predictive value)

```
t<- training.rd[,!sapply(training.rd,function(x) any(is.na(x)))]
t <- t[,-c(1:7)]
nvar <- nearZeroVar(t)
tm <-  t[,-nvar]; dim(tm)
```

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

Test & Validation Data The 19622 rows are splitted. 13737 rows are used to develop the model, 5885 rows are kept aside to validate the model(s)

```
# create training set indexes with 70% of data
inTrain <- createDataPartition(y=tm$classe,p=0.70, list=FALSE)
training.dt <- tm[inTrain,]
val.dt <- tm[-inTrain,]
dim(training.dt); dim(val.dt)
```

```
## [1] 13737    53
```

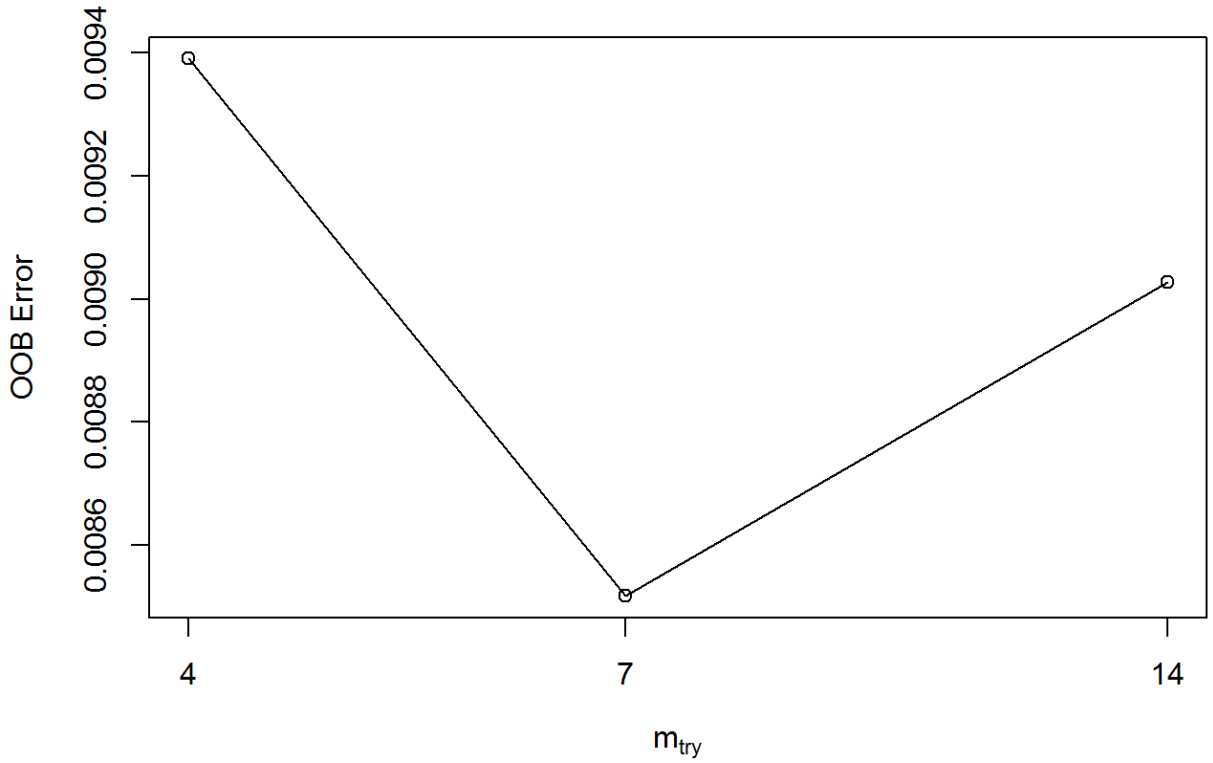
```
## [1] 5885   53
```

Model Development As we have a classification problem (5 classes) a very popular and accurate method is random forest tree (RF) along with boosting. The 2nd method is therefore is gradient boosted Regression Model (GBM). As RF showed very accurate prediction with default values it was the model of choice. I tried to improve the default RF model but was not able to improve it with mtry.

- Model 1 (f1) : Random Forest, default setting
- Model 2 (f2) : Generalized Boosted Regression Modeling
- Model 3 (f3) : Random Forest, with optimal mtry parameter

```
tuneRF(training.dt[,-53], training.dt[,53])
```

```
## mtry = 7  OOB error = 0.85%
## Searching left ...
## mtry = 4     OOB error = 0.94%
## -0.1025641 0.05
## Searching right ...
## mtry = 14    OOB error = 0.9%
## -0.05982906 0.05
```




```
##        mtry    OOBError
## 4.OOB     4 0.009390697
## 7.OOB     7 0.008517143
## 14.OOB   14 0.009026716
```

```
f1 <- randomForest(classe ~ ., data = training.dt)
f2 <- train(classe ~ ., data = training.dt,method="gbm",verbose=FALSE)
f3 <- randomForest(classe ~ ., data = training.dt, mtry=7)
```

Model Validation The 3 models are tested against the training data set and validated against the validation training set

```
pred.t1 <- predict(f1, training.dt)
c.t1<- confusionMatrix(training.dt$classe,pred.t1)

pred.t2 <- predict(f2, training.dt)
c.t2<- confusionMatrix(training.dt$classe,pred.t2)

pred.t3 <- pred.t3 <- predict(f3, training.dt)
c.t3 <- confusionMatrix(training.dt$classe,pred.t3)

pred.v1 <- predict(f1, val.dt)
c.v1 <- confusionMatrix(val.dt$classe,pred.v1)

pred.v2 <- predict(f2, val.dt)
c.v2<- confusionMatrix(val.dt$classe,pred.v2)

pred.v3 <- predict(f3, val.dt)
c.v3 <- confusionMatrix(val.dt$classe,pred.v3)
```

Print Confusion Matrix for Training Data Set

```
c.t1; c.t2;c.t3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     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             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3866   25   11    3    1
##          B   57 2553   47    0    1
##          C    0   51 2322   19    4
##          D    2    2   69 2173    6
##          E    2   10   14   28 2471
##
## Overall Statistics
##
##                Accuracy : 0.9744
##                  95% CI : (0.9716, 0.977)
##     No Information Rate : 0.2859
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9676
##  Mcnemar's Test P-Value : 2.279e-13
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9845   0.9667   0.9428   0.9775   0.9952
## Specificity            0.9959   0.9905   0.9934   0.9931   0.9952
## Pos Pred Value         0.9898   0.9605   0.9691   0.9649   0.9786
## Neg Pred Value         0.9938   0.9921   0.9876   0.9956   0.9989
## Prevalence             0.2859   0.1923   0.1793   0.1618   0.1808
## Detection Rate         0.2814   0.1858   0.1690   0.1582   0.1799
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9902   0.9786   0.9681   0.9853   0.9952
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     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             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

*In Error Sample Rates** t1 (model 1) had no in-errors, t2 (model 2) had about 2.5% in-errors and t3 (model 3) again had no in-errors

Print Confusion Matrix for Validation Data Set

```
c.v1; c.v2;c.v3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    6 1130    3    0    0
##          C    0    7 1016    3    0
##          D    0    0   11  952    1
##          E    0    0    0    3 1079
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9917, 0.9959)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9930   0.9864   0.9937   0.9991
## Specificity            0.9998   0.9981   0.9979   0.9976   0.9994
## Pos Pred Value         0.9994   0.9921   0.9903   0.9876   0.9972
## Neg Pred Value         0.9986   0.9983   0.9971   0.9988   0.9998
## Prevalence             0.2853   0.1934   0.1750   0.1628   0.1835
## Detection Rate         0.2843   0.1920   0.1726   0.1618   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9981   0.9955   0.9922   0.9957   0.9992
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1643   24    7    0    0
##          B   26 1087   23    1    2
##          C    0   34  983    9    0
##          D    3    3   31  925    2
##          E    3   13   11   15 1040
##
## Overall Statistics
##
##                Accuracy : 0.9648
##                  95% CI : (0.9598, 0.9694)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9555
##  Mcnemar's Test P-Value : 1.165e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9809   0.9363   0.9318   0.9737   0.9962
## Specificity            0.9926   0.9890   0.9911   0.9921   0.9913
## Pos Pred Value         0.9815   0.9543   0.9581   0.9595   0.9612
## Neg Pred Value         0.9924   0.9844   0.9852   0.9949   0.9992
## Prevalence             0.2846   0.1973   0.1793   0.1614   0.1774
## Detection Rate         0.2792   0.1847   0.1670   0.1572   0.1767
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9868   0.9626   0.9614   0.9829   0.9937
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    6 1130    3    0    0
##          C    0   10 1013    3    0
##          D    0    0   11  952    1
##          E    0    0    0    3 1079
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9911, 0.9954)
##     No Information Rate : 0.2853
##     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            0.9964   0.9904   0.9864   0.9937   0.9991
## Specificity            0.9998   0.9981   0.9973   0.9976   0.9994
## Pos Pred Value         0.9994   0.9921   0.9873   0.9876   0.9972
## Neg Pred Value         0.9986   0.9977   0.9971   0.9988   0.9998
## Prevalence             0.2853   0.1939   0.1745   0.1628   0.1835
## Detection Rate         0.2843   0.1920   0.1721   0.1618   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9981   0.9942   0.9918   0.9957   0.9992
```

Out Error Rates out of sample errors The confusion matrix for classifying the validation data, using the models developed with the training data sets showd an accuracy of 99.41% (Model 1), 96.48% (Model 2) and 99.34% (Model 3).

Prediction f1 - Model 1, (the model with the lowest error rate related to the validation set is used to predict the classifications for the test data set (20 observation) The model was able to predict the outcome correctly(no errors)

```
pred.r <- predict(f1, testing.rd)
pred.r
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

```
# code for submitting the predictions commented out
#answers = rep("X", 20)
#  pml_write_files = function(x){
#    n = length(x)
#    for(i in 1:n){
#      filename = paste0("problem_id_",i,".txt")
#      write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FAL
SE)
#    }
#  }

#  pml_write_files(pred.r)
```

Appendix

**Some Model 1 insight

```
summary(f1)
```

```
##                 Length Class  Mode
## call               3  -none- call
## type               1  -none- character
## predicted      13737  factor numeric
## err.rate        3000  -none- numeric
## confusion         30  -none- numeric
## votes          68685  matrix numeric
## oob.times      13737  -none- numeric
## classes            5  -none- character
## importance        52  -none- numeric
## importanceSD       0  -none- NULL
## localImportance    0  -none- NULL
## proximity          0  -none- NULL
## ntree              1  -none- numeric
## mtry               1  -none- numeric
## forest            14  -none- list
## y              13737  factor numeric
## test               0  -none- NULL
## inbag              0  -none- NULL
## terms              3  terms  call
```

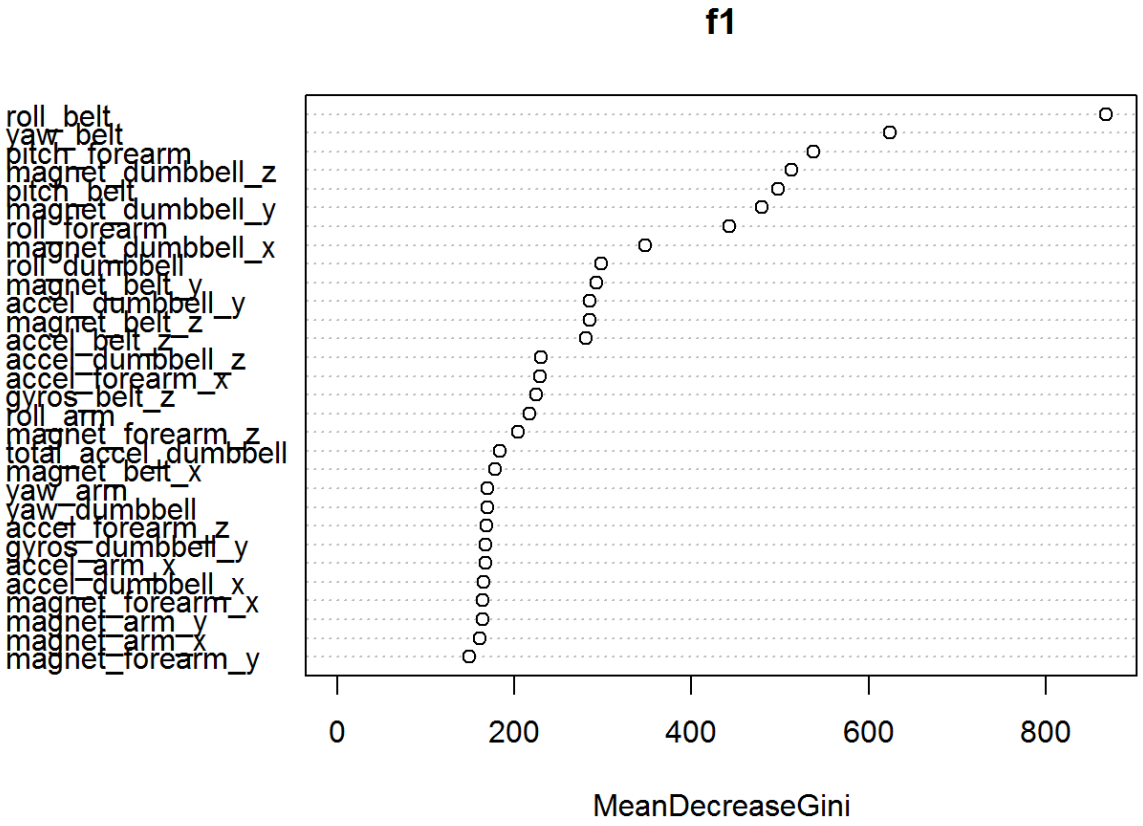
```
str(f1)
```

```
## List of 19
##  $ call           : language randomForest(formula = classe ~ ., data = train
ing.dt)
##  $ type           : chr "classification"
##  $ predicted      : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1
1 1 ...
##   ..- attr(*, "names")= chr [1:13737] "1" "2" "4" "5" ...
##  $ err.rate       : num [1:500, 1:6] 0.0738 0.1003 0.0864 0.0807 0.0745 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:6] "OOB" "A" "B" "C" ...
##  $ confusion      : num [1:5, 1:6] 3903 16 0 0 0 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:5] "A" "B" "C" "D" ...
##   .. ..$ : chr [1:6] "A" "B" "C" "D" ...
##  $ votes          : matrix [1:13737, 1:5] 1 1 1 1 0.994 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:13737] "1" "2" "4" "5" ...
##   .. ..$ : chr [1:5] "A" "B" "C" "D" ...
##   ..- attr(*, "class")= chr [1:2] "matrix" "votes"
##  $ oob.times      : num [1:13737] 185 189 182 189 170 208 179 207 184 20
2 ...
##  $ classes        : chr [1:5] "A" "B" "C" "D" ...
##  $ importance     : num [1:52, 1] 867.3 497.8 623.5 144.1 68.9 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:52] "roll_belt" "pitch_belt" "yaw_belt" "total_accel_bel
t" ...
##   .. ..$ : chr "MeanDecreaseGini"
##  $ importanceSD   : NULL
##  $ localImportance: NULL
##  $ proximity      : NULL
##  $ ntree          : num 500
##  $ mtry           : num 7
##  $ forest         :List of 14
##   ..$ ndbigtree : int [1:500] 1297 1821 1379 1291 1381 1497 1501 1455 1641 1
475 ...
##   ..$ nodestatus: int [1:1885, 1:500] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ bestvar   : int [1:1885, 1:500] 7 1 52 41 34 14 50 19 3 50 ...
##   ..$ treemap   : int [1:1885, 1:2, 1:500] 2 4 6 8 10 12 14 16 18 20 ...
##   ..$ nodepred  : int [1:1885, 1:500] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ xbestsplit: num [1:1885, 1:500] 0.06 130.5 93.5 -33.55 -49.5 ...
##   ..$ pid       : num [1:5] 1 1 1 1 1
##   ..$ cutoff    : num [1:5] 0.2 0.2 0.2 0.2 0.2
##   ..$ ncat      : Named int [1:52] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..- attr(*, "names")= chr [1:52] "roll_belt" "pitch_belt" "yaw_belt" "t
otal_accel_belt" ...
##   ..$ maxcat    : int 1
##   ..$ nrnodes   : int 1885
```

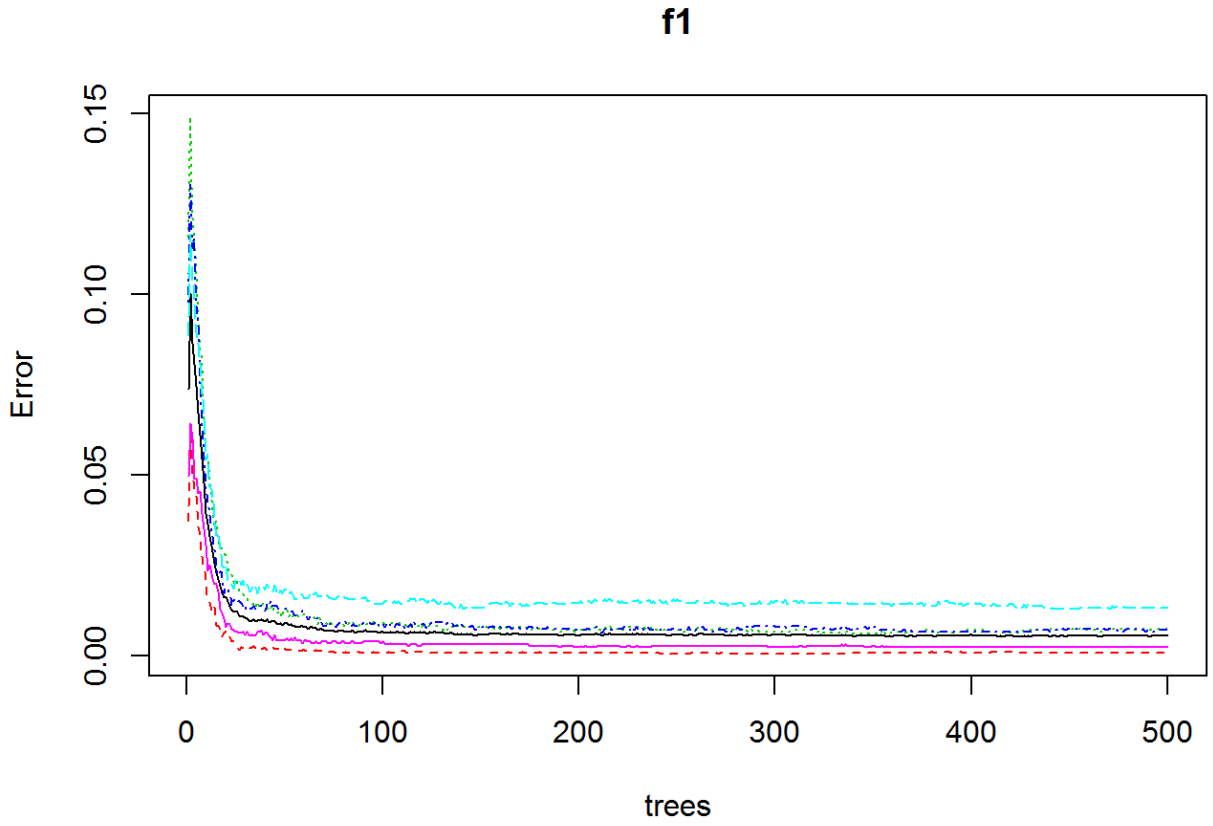
```
##   ..$ ntree     : num 500
##   ..$ nclass    : int 5
##   ..$ xlevels   :List of 52
##   .. ..$ roll_belt           : num 0
##   .. ..$ pitch_belt          : num 0
##   .. ..$ yaw_belt            : num 0
##   .. ..$ total_accel_belt    : num 0
##   .. ..$ gyros_belt_x        : num 0
##   .. ..$ gyros_belt_y        : num 0
##   .. ..$ gyros_belt_z        : num 0
##   .. ..$ accel_belt_x        : num 0
##   .. ..$ accel_belt_y        : num 0
##   .. ..$ accel_belt_z        : num 0
##   .. ..$ magnet_belt_x       : num 0
##   .. ..$ magnet_belt_y       : num 0
##   .. ..$ magnet_belt_z       : num 0
##   .. ..$ roll_arm            : num 0
##   .. ..$ pitch_arm           : num 0
##   .. ..$ yaw_arm             : num 0
##   .. ..$ total_accel_arm     : num 0
##   .. ..$ gyros_arm_x         : num 0
##   .. ..$ gyros_arm_y         : num 0
##   .. ..$ gyros_arm_z         : num 0
##   .. ..$ accel_arm_x         : num 0
##   .. ..$ accel_arm_y         : num 0
##   .. ..$ accel_arm_z         : num 0
##   .. ..$ magnet_arm_x        : num 0
##   .. ..$ magnet_arm_y        : num 0
##   .. ..$ magnet_arm_z        : num 0
##   .. ..$ roll_dumbbell       : num 0
##   .. ..$ pitch_dumbbell      : num 0
##   .. ..$ yaw_dumbbell        : num 0
##   .. ..$ total_accel_dumbbell: num 0
##   .. ..$ gyros_dumbbell_x    : num 0
##   .. ..$ gyros_dumbbell_y    : num 0
##   .. ..$ gyros_dumbbell_z    : num 0
##   .. ..$ accel_dumbbell_x    : num 0
##   .. ..$ accel_dumbbell_y    : num 0
##   .. ..$ accel_dumbbell_z    : num 0
##   .. ..$ magnet_dumbbell_x   : num 0
##   .. ..$ magnet_dumbbell_y   : num 0
##   .. ..$ magnet_dumbbell_z   : num 0
##   .. ..$ roll_forearm        : num 0
##   .. ..$ pitch_forearm       : num 0
##   .. ..$ yaw_forearm         : num 0
##   .. ..$ total_accel_forearm : num 0
##   .. ..$ gyros_forearm_x     : num 0
##   .. ..$ gyros_forearm_y     : num 0
##   .. ..$ gyros_forearm_z     : num 0
```

```
##   .. ..$ accel_forearm_x     : num 0
##   .. ..$ accel_forearm_y     : num 0
##   .. ..$ accel_forearm_z     : num 0
##   .. ..$ magnet_forearm_x    : num 0
##   .. ..$ magnet_forearm_y    : num 0
##   .. ..$ magnet_forearm_z    : num 0
##  $ y              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1
1 1 ...
##   ..- attr(*, "names")= chr [1:13737] "1" "2" "4" "5" ...
##  $ test           : NULL
##  $ inbag          : NULL
##  $ terms          :Classes 'terms', 'formula' length 3 classe ~ roll_belt +
pitch_belt + yaw_belt + total_accel_belt + gyros_belt_x +      gyros_belt_y + g
yros_belt_z + accel_belt_x + accel_belt_y + accel_belt_z +  ...
##   .. ..- attr(*, "variables")= language list(classe, roll_belt, pitch_belt,
yaw_belt, total_accel_belt, gyros_belt_x,      gyros_belt_y, gyros_belt_z, acce
l_belt_x, accel_belt_y, accel_belt_z,  ...
##   .. ..- attr(*, "factors")= int [1:53, 1:52] 0 1 0 0 0 0 0 0 0 0 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:53] "classe" "roll_belt" "pitch_belt" "yaw_belt" ...
##   .. .. .. ..$ : chr [1:52] "roll_belt" "pitch_belt" "yaw_belt" "total_accel
_belt" ...
##   .. ..- attr(*, "term.labels")= chr [1:52] "roll_belt" "pitch_belt" "yaw_be
lt" "total_accel_belt" ...
##   .. ..- attr(*, "order")= int [1:52] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..- attr(*, "intercept")= num 0
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(classe, roll_belt, pitch_belt, y
aw_belt, total_accel_belt, gyros_belt_x,      gyros_belt_y, gyros_belt_z, accel
_belt_x, accel_belt_y, accel_belt_z,  ...
##   .. ..- attr(*, "dataClasses")= Named chr [1:53] "factor" "numeric" "numeri
c" "numeric" ...
##   .. .. ..- attr(*, "names")= chr [1:53] "classe" "roll_belt" "pitch_belt"
"yaw_belt" ...
##  - attr(*, "class")= chr [1:2] "randomForest.formula" "randomForest"
```

```
varImpPlot(f1, sort=TRUE)
```


f1
roll_belt
yaw_belt
pitch_forearm
magnet_dumbbell_z
pitch_belt
magnet_dumbbell_y
roll_forearm
magnet_dumbbell_x
roll_dumbbell
magnet_belt_y
accel_dumbbell_y
magnet_belt_z
accel_belt_z
accel_dumbbell_z
accel_forearm_x
gyros_belt_z
roll_arm
magnet_forearm_z
total_accel_dumbbell
magnet_belt_x
yaw_arm
yaw_dumbbell
accel_forearm_z
gyros_dumbbell_y
accel_arm_x
accel_dumbbell_x
magnet_forearm_x
magnet_arm_y
magnet_arm_x
magnet_forearm_y
0
200
400
600
800
MeanDecreaseGini

```
plot(f1)
```


f1
Error
0.00
0.05
0.10
0.15
0
100
200
300
400
500
trees