Prediction Assignment Writeup

Serg C

November 26, 2017

Machine Learning, Coursera, Johns Hopkins University

Synopsis

Human Activity Recognition (HAR) has emerged as a key research area in the last years and is gaining increasing interest due to many potential applications for HAR, like: elderly monitoring, life log systems for monitoring energy expenditure and for supporting weight-loss programs, and digital assistants for weight lifting exercises.

This (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) research was focused on investigating “how (well)” some activity is being performed by 6 participants who were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The data from multiple sensors were collected in the dataset.

The goal of this project is to predict the manner or “how well” the participants did the exercise using data from accelerometers on the belt, forearm, arm, and dumbell.

Data

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

Data Preparation

1. Download and read the dataset

```
wdir <- '.'
if(!file.exists("pml-training.csv")){
  fil <- paste(wdir,'pml-training.csv',sep="/")
  fUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
  download.file( fUrl, fil )
}
if(!file.exists("pml-testing.csv")){
  fil <- paste(wdir,'pml-testing.csv',sep="/")
  fUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
  download.file( fUrl, fil )
}
pml.training <- read.csv( paste( wdir,"pml-training.csv",sep="/") )
pml.testing  <- read.csv( paste( wdir,"pml-testing.csv",sep="/") )
```

2. Extract accelerometers data then partition it into train, test and validation sets

Extract all accelerometer variables plus 'classe' from both datasets. Carve a validation partition off the training data set. Basic data exploration.

```
training<- pml.training[,c(160,grep('^accel',names(pml.training)))]
testing <- pml.testing [,c(160,grep('^accel',names(pml.training)))]

inTrn <- createDataPartition(y=training$classe, p=0.5, list=FALSE)
### inTrn <- createDataPartition(y=training$classe, p=0.1, list=FALSE)
train <- training[ inTrn,]
inVal <- createDataPartition(y=training[-inTrn,]$classe, p=0.8, list=FALSE)
vldt  <- training[ inVal,]
test  <- training[-inVal,]
```

Here is a summary of the datasets for model building

```
dim(train);dim(vldt);dim(test)
```

```
## [1] 9812   13
```

```
## [1] 7850   13
```

```
## [1] 11772    13
```

```
str(train)
```

```
## 'data.frame':    9812 obs. of  13 variables:
##  $ classe          : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ accel_belt_x    : int  -22 -20 -22 -22 -22 -21 -21 -21 -21 -22 ...
##  $ accel_belt_y    : int  4 5 3 4 4 2 4 4 5 5 ...
##  $ accel_belt_z    : int  22 23 21 21 21 22 23 22 21 22 ...
##  $ accel_arm_x     : int  -290 -289 -289 -289 -288 -289 -289 -289 -288 -289 ...
##  $ accel_arm_y     : int  110 110 111 111 111 111 109 110 108 109 ...
##  $ accel_arm_z     : int  -125 -126 -125 -124 -124 -124 -121 -122 -124 -125 ...
##  $ accel_dumbbell_x: int  -233 -232 -232 -234 -234 -234 -233 -233 -231 -234 ...
##  $ accel_dumbbell_y: int  47 46 47 46 48 47 48 47 47 46 ...
##  $ accel_dumbbell_z: int  -269 -270 -270 -272 -268 -270 -271 -272 -268 -272 ...
##  $ accel_forearm_x : int  192 196 195 193 193 192 194 192 192 193 ...
##  $ accel_forearm_y : int  203 204 205 205 202 201 204 204 206 205 ...
##  $ accel_forearm_z : int  -216 -213 -215 -213 -214 -214 -215 -213 -216 -215 ...
```

Model Selection

Here we will fit several models and then choose the one based on out-of-sample accuracy.

1. Boosting Tree model with cross-validation

```
set.seed( 12345 )
fit1<-train( classe~., data=train, method="gbm", verbose=FALSE,
              trControl = trainControl(method="cv", number = 5))
```

```
## Loading required package: gbm
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

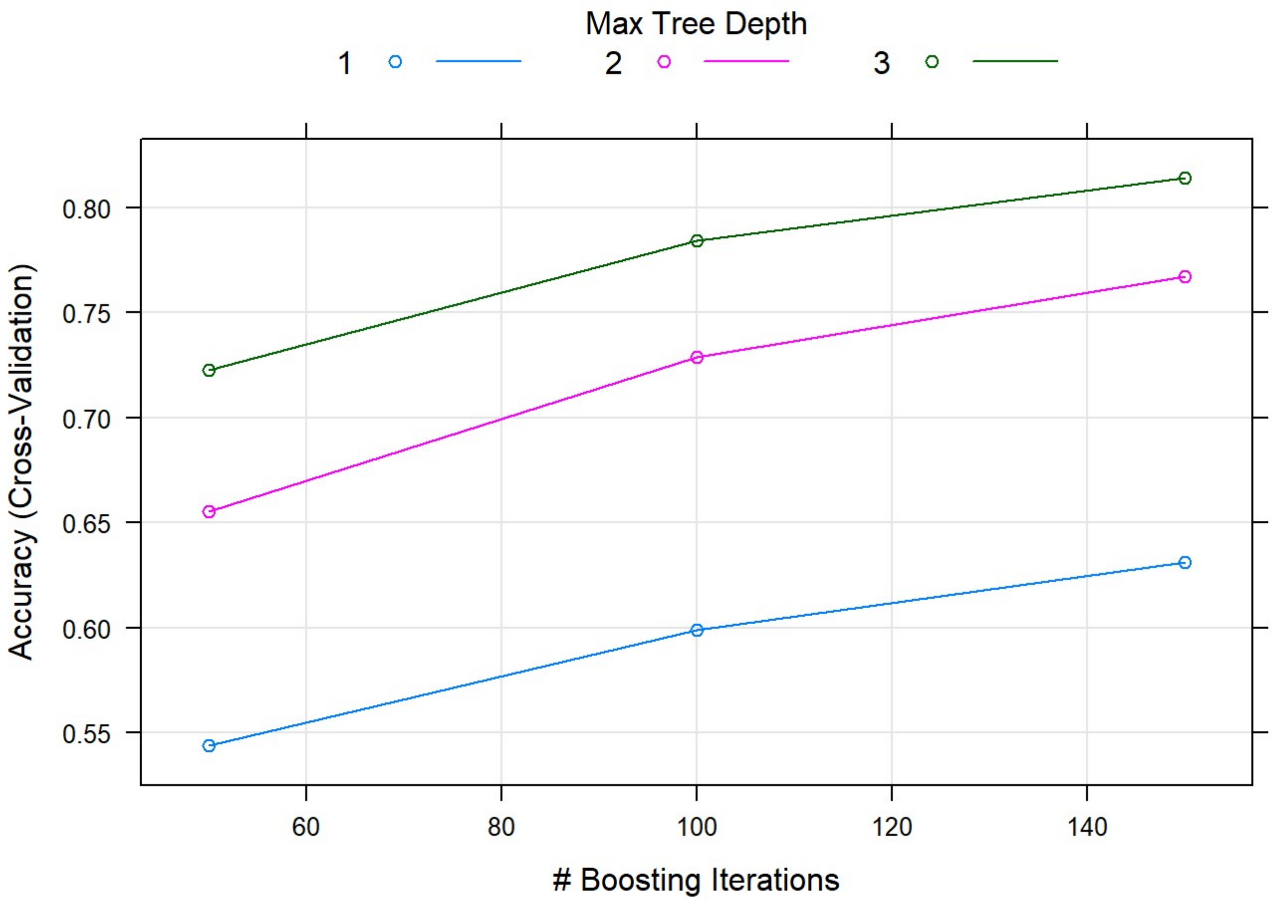
```
## Loaded gbm 2.1.3
```

```
## Loading required package: plyr
```

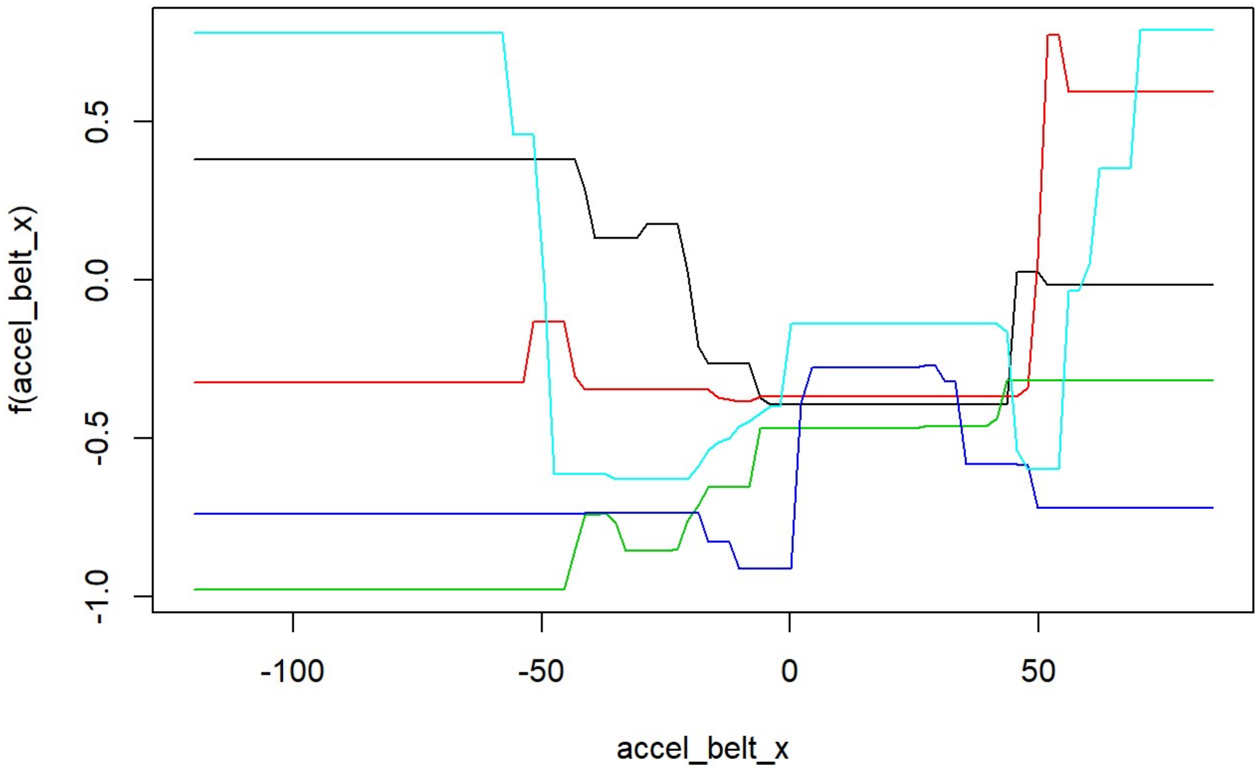
```
print( fit1 )
```

```
## Stochastic Gradient Boosting
##
## 9812 samples
##   12 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 7850, 7849, 7850, 7849, 7850
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.5437246  0.4138052
##   1                  100      0.5986570  0.4864122
##   1                  150      0.6311673  0.5293833
##   2                   50      0.6553213  0.5600810
##   2                  100      0.7288028  0.6550536
##   2                  150      0.7670201  0.7041266
##   3                   50      0.7227879  0.6472472
##   3                  100      0.7844457  0.7262403
##   3                  150      0.8140018  0.7640147
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
plot( fit1 )
```




```
plot(fit1$finalModel)
```




```
prd1 <- predict(fit1, vldt)
confusionMatrix(prd1, vldt$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3996  313   29    0    0
##          B   77 2303   44    0    0
##          C  157  245  272    0    0
##          D  224  105    1    0    0
##          E    7   77    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.8371
##                  95% CI : (0.8287, 0.8452)
##     No Information Rate : 0.5683
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7103
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8958   0.7568  0.78613       NA       NA
## Specificity            0.8991   0.9748  0.94643  0.95796   0.9893
## Pos Pred Value         0.9212   0.9501  0.40356       NA       NA
## Neg Pred Value         0.8676   0.8636  0.98969       NA       NA
## Prevalence             0.5683   0.3876  0.04408  0.00000   0.0000
## Detection Rate         0.5090   0.2934  0.03465  0.00000   0.0000
## Detection Prevalence   0.5526   0.3088  0.08586  0.04204   0.0107
## Balanced Accuracy      0.8974   0.8658  0.86628       NA       NA
```

Now we will try to tune over the number/complexity of trees.

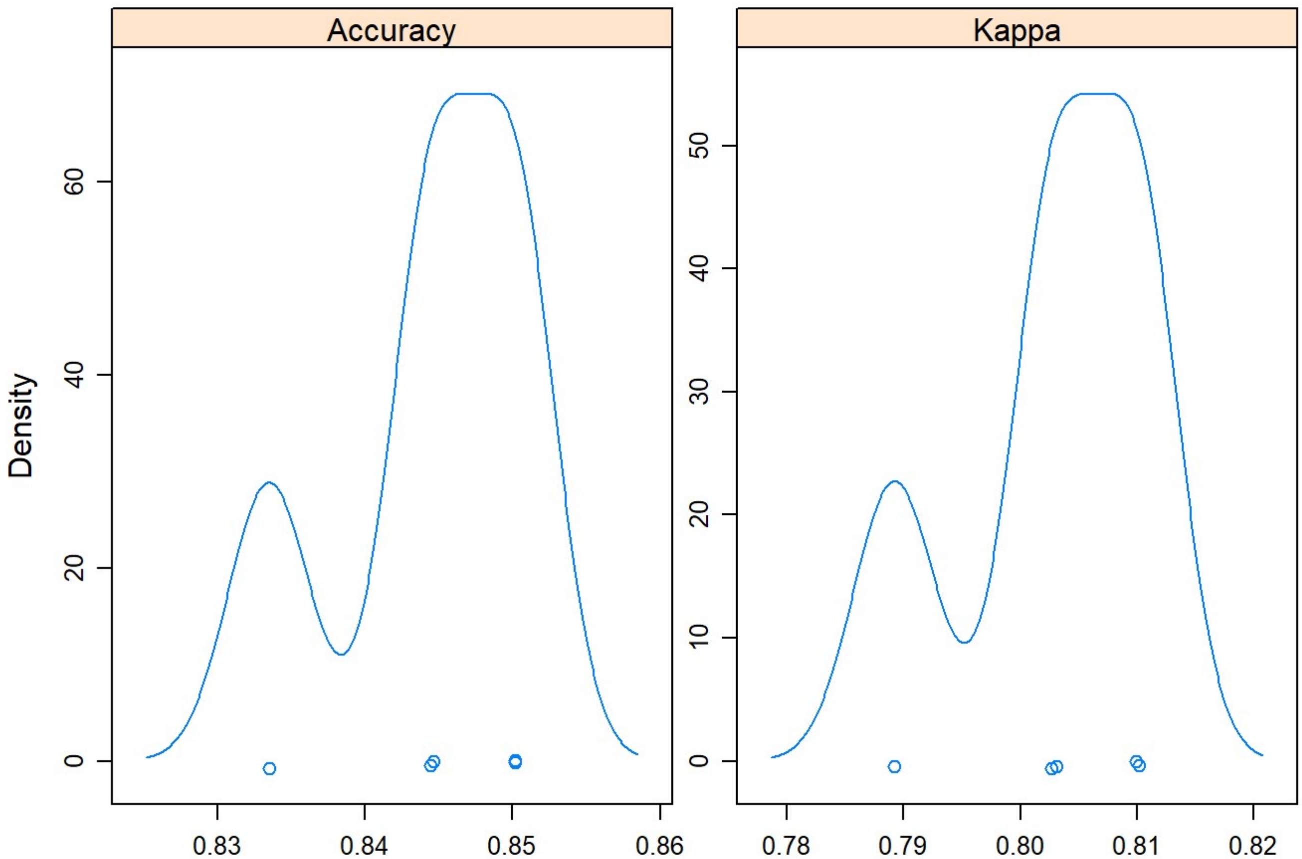
```
f1Grid <- expand.grid(.interaction.depth=(1:3)*2,
 .n.trees=(1:5)*20, .shrinkage=.1, .n.minobsinnode = c(10))

fit1<-train( classe~., data=train, method="gbm", verbose=FALSE,
              trControl = trainControl(method="cv", number = 5),
              tuneGrid=f1Grid)
prd1 <- predict(fit1, vldt)
confusionMatrix(prd1, vldt$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4082  249   20    0    0
##          B   77 2442   37    0    0
##          C  122  201  289    0    0
##          D  171   88    0    0    0
##          E    9   63    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.8679
##                  95% CI : (0.8602, 0.8753)
##     No Information Rate : 0.5683
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7621
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9150   0.8025  0.83526       NA       NA
## Specificity            0.9206   0.9763  0.95696  0.96701 0.990828
## Pos Pred Value         0.9382   0.9554  0.47222       NA       NA
## Neg Pred Value         0.8917   0.8865  0.99212       NA       NA
## Prevalence             0.5683   0.3876  0.04408  0.00000 0.000000
## Detection Rate         0.5200   0.3111  0.03682  0.00000 0.000000
## Detection Prevalence   0.5543   0.3256  0.07796  0.03299 0.009172
## Balanced Accuracy      0.9178   0.8894  0.89611       NA       NA
```

The accuracy of the tuned model can be visualized as follws.

```
resampleHist(fit1)
```



2. Random Forests model

Here we fit a random forest model with three fold cross validation. Caret will use cross validation to select best predictors.

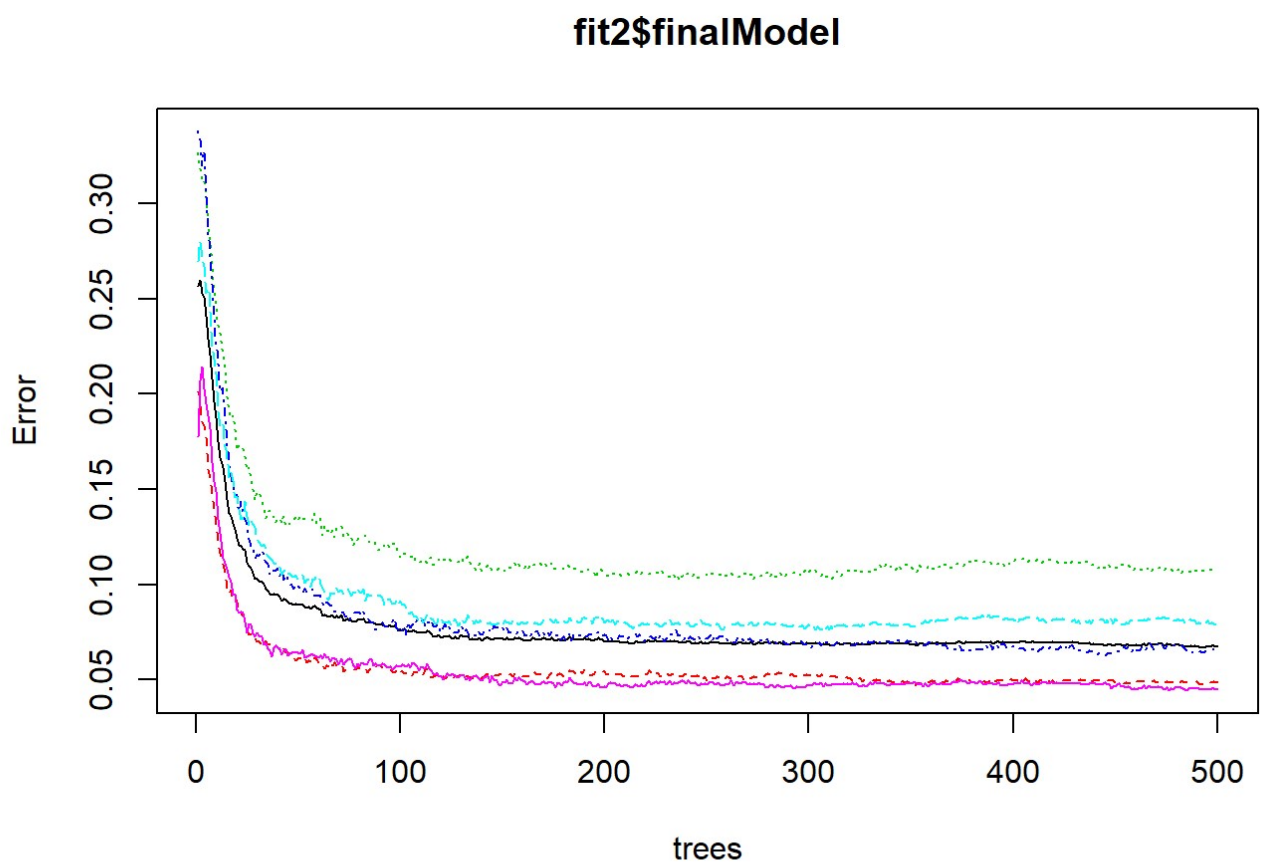
```
fit2<-train( classe~., data=train, method="rf" , importance = T,
               trControl = trainControl(method="cv", number = 5 ))
varImp(fit2)
```

```
## rf variable importance
##
##   variables are sorted by maximum importance across the classes
##                        A     B     C     D       E
## accel_belt_z     29.376 50.67 55.57 28.37 100.000
## accel_forearm_x  28.188 45.84 48.54 76.30  31.612
## accel_dumbbell_y 45.118 67.87 73.73 50.97  42.274
## accel_dumbbell_z 25.851 37.22 69.80 43.14  39.514
## accel_arm_x      60.385 28.24 43.51 61.12  16.343
## accel_dumbbell_x 23.026 49.27 54.28 24.80  24.738
## accel_forearm_z  16.643 29.34 52.11 19.13  12.358
## accel_forearm_y  15.608 25.23 41.81 16.76   9.418
## accel_arm_z      14.627 24.48 35.56 19.53   8.573
## accel_arm_y      17.075 30.98 24.99 26.77  20.014
## accel_belt_x     13.502 25.16 24.80 12.58   5.252
## accel_belt_y      5.348 22.23 17.49 21.01   0.000
```

```
print( fit2 )
```

```
## Random Forest
##
## 9812 samples
##   12 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 7850, 7849, 7850, 7850, 7849
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9237670  0.9035395
##    7    0.9169389  0.8949159
##   12    0.9020594  0.8760846
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

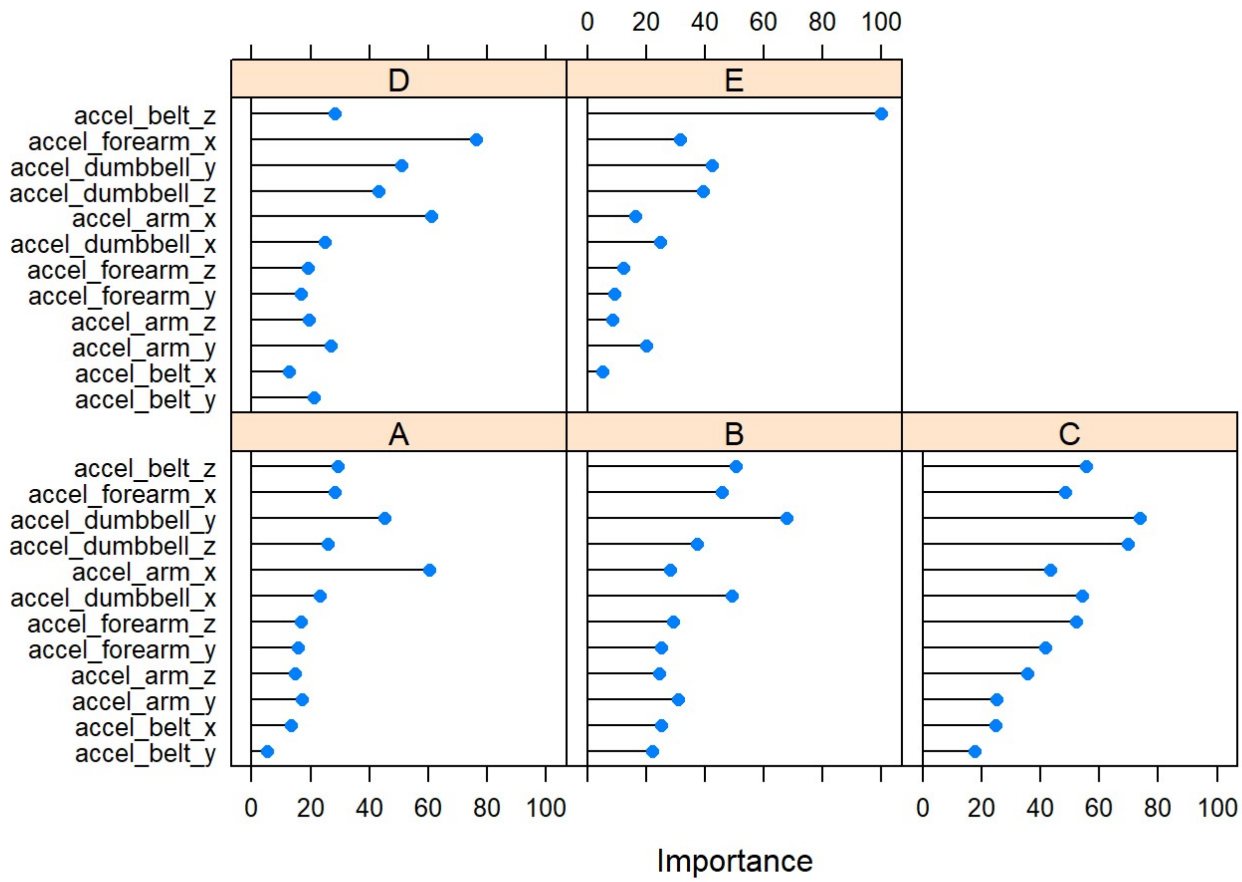
```
plot(fit2$finalModel)
```




```
prd2 <- predict(fit2,vldt)
confusionMatrix(prd2, vldt$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4370   65   12    0    0
##          B   22 2900   12    0    0
##          C   32   50  321    0    0
##          D   34   14    1    0    0
##          E    3   14    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.967
##                  95% CI : (0.9628, 0.9708)
##     No Information Rate : 0.5683
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9379
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9796   0.9530  0.92775       NA       NA
## Specificity            0.9773   0.9929  0.98907 0.993758 0.997834
## Pos Pred Value         0.9827   0.9884  0.79653       NA       NA
## Neg Pred Value         0.9733   0.9709  0.99664       NA       NA
## Prevalence             0.5683   0.3876  0.04408 0.000000 0.000000
## Detection Rate         0.5567   0.3694  0.04089 0.000000 0.000000
## Detection Prevalence   0.5665   0.3738  0.05134 0.006242 0.002166
## Balanced Accuracy      0.9784   0.9730  0.95841       NA       NA
```

```
plot(varImp(fit2))
```



3. Bagging

The default settings will be used for treebag method.

```
fit3<-train( classe~., data=train, method="treebag")
```

```
## Loading required package: ipred
```

```
## Loading required package: e1071
```

```
print( fit3 )
```

```
## Bagged CART
##
## 9812 samples
##   12 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 9812, 9812, 9812, 9812, 9812, 9812, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8726568  0.8387704
```

```
prd3 <- predict(fit3,vldt)
confusionMatrix(prd3, vldt$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4305   60   11    0    0
##          B   52 2857   13    0    0
##          C   44   62  317    0    0
##          D   53   34    4    0    0
##          E    7   30    1    0    0
##
## Overall Statistics
##
##                Accuracy : 0.9527
##                  95% CI : (0.9478, 0.9573)
##     No Information Rate : 0.5683
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9119
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9650   0.9389  0.91618       NA       NA
## Specificity            0.9790   0.9865  0.98587  0.98841 0.995159
## Pos Pred Value         0.9838   0.9778  0.74941       NA       NA
## Neg Pred Value         0.9551   0.9623  0.99610       NA       NA
## Prevalence             0.5683   0.3876  0.04408  0.00000 0.000000
## Detection Rate         0.5484   0.3639  0.04038  0.00000 0.000000
## Detection Prevalence   0.5575   0.3722  0.05389  0.01159 0.004841
## Balanced Accuracy      0.9720   0.9627  0.95103       NA       NA
```

```
varImp(fit3)
```

```
## Loading required package: rpart
```

```
## treebag variable importance
##
##                  Overall
## accel_dumbbell_y 100.000
## accel_dumbbell_z  91.930
## accel_belt_z      81.984
## accel_arm_x       71.204
## accel_dumbbell_x  65.146
## accel_forearm_x   36.586
## accel_arm_y       34.309
## accel_forearm_z   26.352
## accel_arm_z       13.233
## accel_belt_x      12.767
## accel_belt_y       9.378
## accel_forearm_y    0.000
```

Compare models

To summup the results accuracy of the models is printed in a table. It looks like the random forest model has better accuracy.

```
##                gbm        rf   treebag
## Accuracy 0.8678981 0.9670064 0.9527389
## Kappa    0.7621353 0.9378561 0.9119156
```