# Cross validation

Gianni Enas

2023-09-14

## Training exercises

The project consists in predicting the the type of exercise performed with the data collected from many sensors fixed all over the moving parts of the bodies and is stored in a variable called "classe" in the training set of the database.

Let's load the required libraries first:

```r
library(caret)
library(gbm)
library(randomForest)
library(tidyverse)
```

And also the test and training set:

```r
pml_training <- read.csv("pml-training.csv")
pml_testing <- read.csv("pml-testing.csv")
```

The dataset contains many variables, many of which are not significant, I will remove some of them that have no evident meaning and then use some caret functions to clean the data getting rid of variables with zero variance ,n/a, or high correlated. First let's remove the user_names, the windows variables and the data variables, then the zero variance and the N/A values

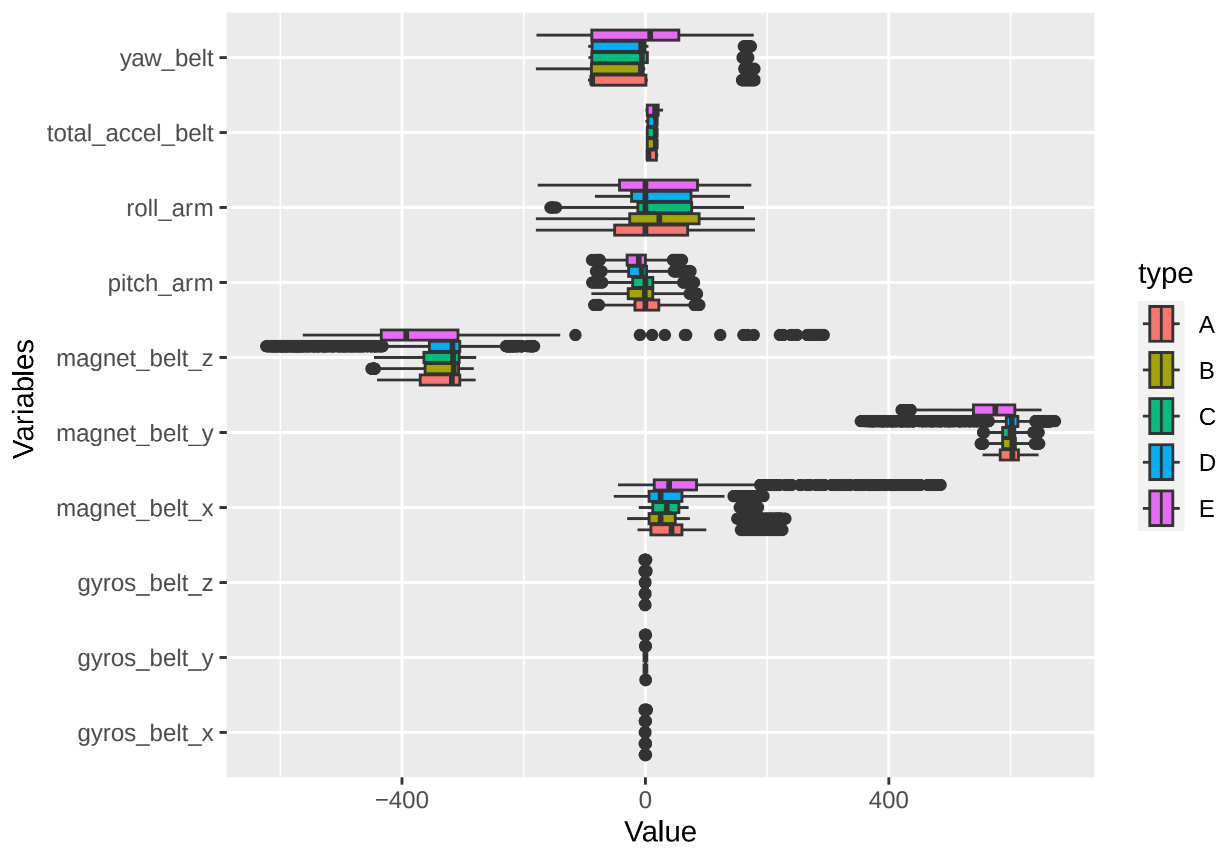
```r
training <- pml_training[, -c(1,2,3,4,5,6,7)]
testing <- pml_testing[, -c(1,2,3,4,5,6,7)]
zero_values <- nearZeroVar(training)
training <- training[, - zero_values]
testing <- testing[, -zero_values]
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
rm(pml_training, pml_testing)
```

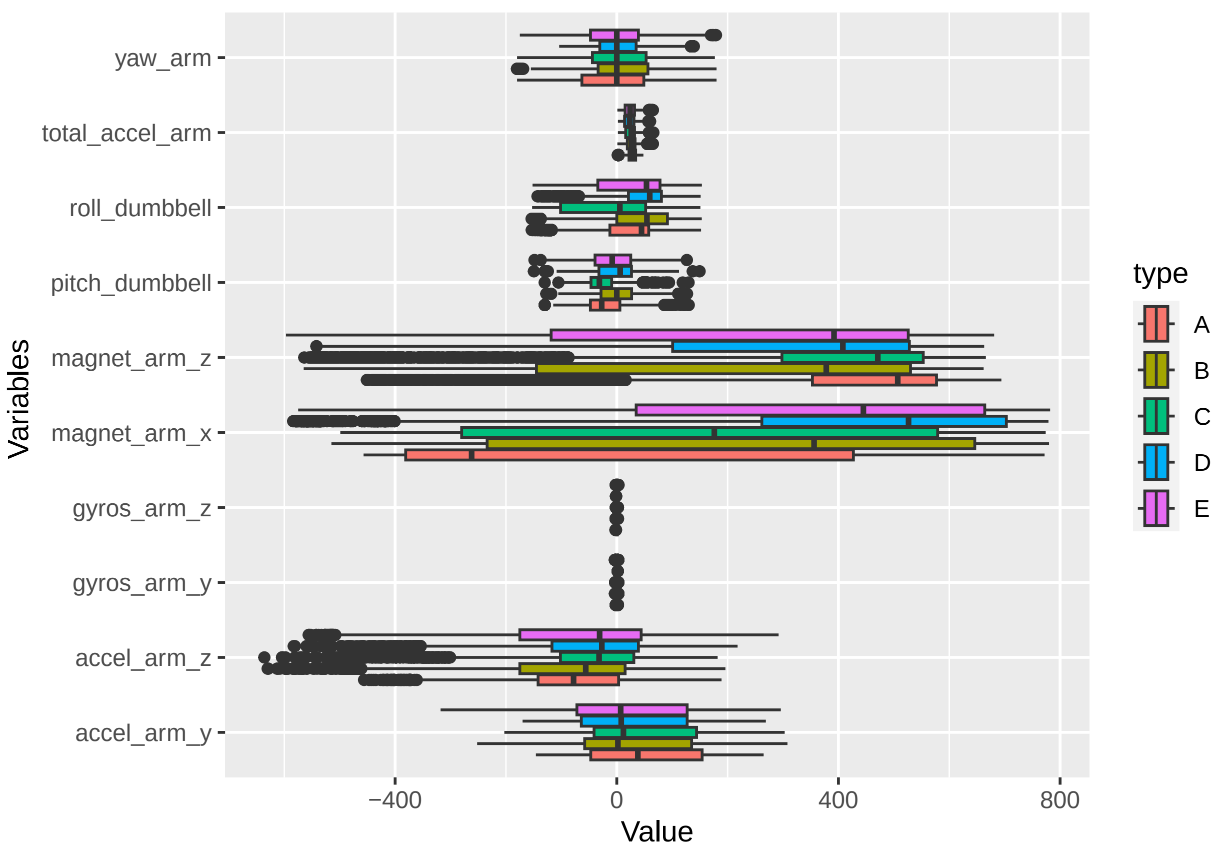
Let's now get rid of the variables with correlation $> 0.8$

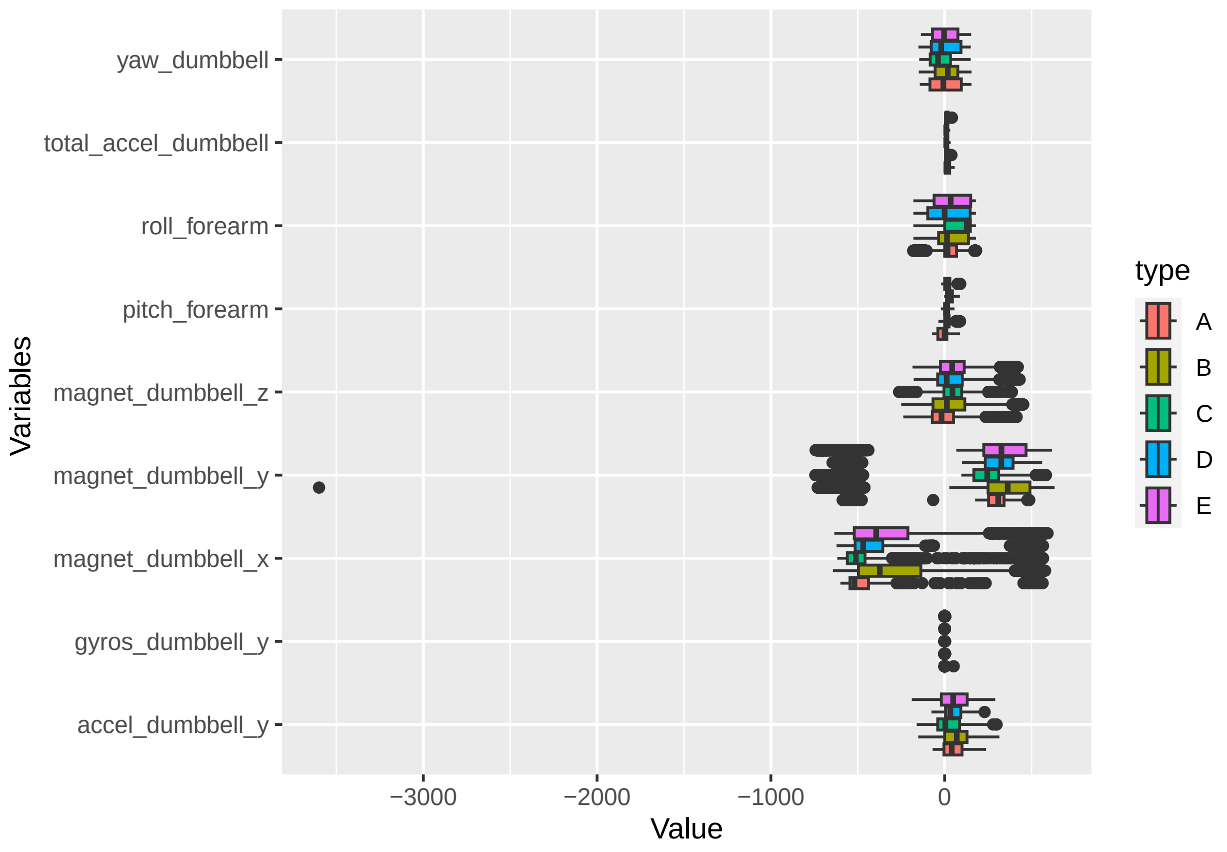
```r
training$classe = as.factor(training$classe)
correlated <- findCorrelation(cor(training[,1:52]), cutoff = 0.8)
training <- training[,-correlated]
testing <- testing[,-correlated]
```

## Plotting the data

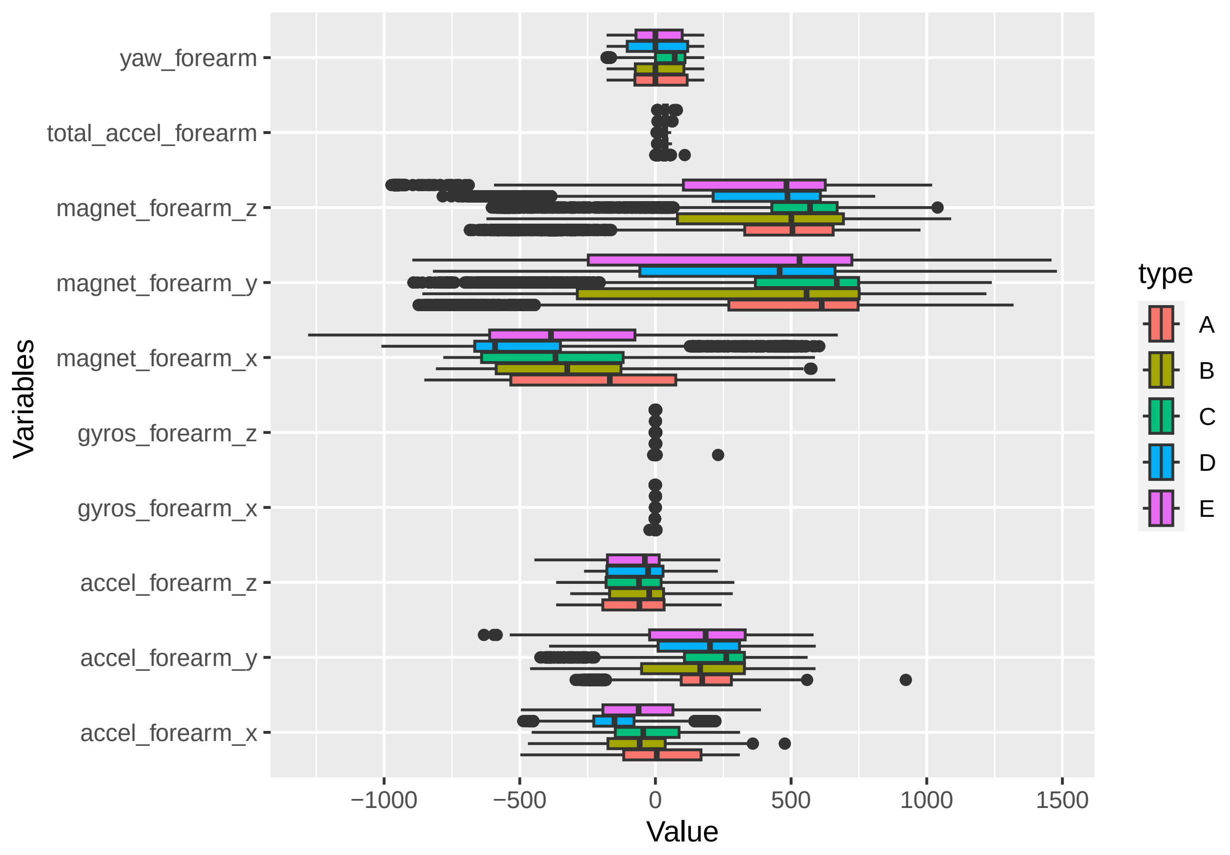
Let's do some plotting with the remaining variables to check their values.


yaw_belt
total_accel_belt
roll_arm
pitch_arm
magnet_belt_z
magnet_belt_y
magnet_belt_x
gyros_belt_z
gyros_belt_y
gyros_belt_x
Variables
−400
0
400
Value
type
A
B
C
D
E


yaw_arm
total_accel_arm
roll_dumbbell
pitch_dumbbell
magnet_arm_z
magnet_arm_x
gyros_arm_z
gyros_arm_y
accel_arm_z
accel_arm_y
Variables
−400
0
400
800
Value
type
A
B
C
D
E



Variables
yaw_dumbbell
total_accel_dumbbell
roll_forearm
pitch_forearm
magnet_dumbbell_z
magnet_dumbbell_y
magnet_dumbbell_x
gyros_dumbbell_y
accel_dumbbell_y
−3000
−2000
−1000
0
Value
type
A
B
C
D
E



As we can deduce from the plots there are few variables which values are zero or close to zero and by my assessment they are not significant therefore is better to remove them and shrink the data further.

```
training <- subset(training, select = -c(gyros_belt_x, gyros_belt_y, gyros_belt_z, gyros_arm_z, gyros_a
testing <- subset(testing, select = -c(gyros_belt_x, gyros_belt_y, gyros_belt_z, gyros_arm_z, gyros_arm
```

Let's fit now two different models to compare and do cross validation

```
training$classe = as.factor(training$classe)
testing = testing[,-31]
control <- trainControl(method = "cv", number = 5)

fitgbm <- train(classe~., data = training, method = "gbm", trControl = control, verbose = FALSE)
fitrf <- train(classe~., data = training, method = "rf", trControl = control)

fitgbm
```

```
## Stochastic Gradient Boosting
##
## 19622 samples
##    30 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 15698, 15698, 15697, 15696, 15699
## Resampling results across tuning parameters:
##
```

```
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7115983  0.6344533
##   1                  100      0.7844764  0.7272862
##   1                  150      0.8189786  0.7708894
##   2                   50      0.8323820  0.7876593
##   2                  100      0.8862501  0.8560146
##   2                  150      0.9155029  0.8930758
##   3                   50      0.8759556  0.8429551
##   3                  100      0.9250842  0.9051783
##   3                  150      0.9464379  0.9322232
##
## 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.
```

```
fitrf
```

```
## Random Forest
##
## 19622 samples
##    30 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 15699, 15697, 15697, 15698, 15697
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9903680  0.9878149
##   16    0.9915401  0.9892981
##   30    0.9847618  0.9807206
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 16.
```

```
confusionMatrix(fitgbm)
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 27.9  0.8  0.0  0.1  0.0
##          B  0.3 17.6  1.0  0.1  0.3
##          C  0.1  0.7 16.2  0.7  0.2
##          D  0.1  0.1  0.3 15.4  0.4
##          E  0.0  0.1  0.0  0.1 17.5
##
##  Accuracy (average) : 0.9464
```

```
confusionMatrix(fitrf)
```

```
## 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 19.1  0.2  0.0  0.0
##          C  0.0  0.1 17.2  0.2  0.0
##          D  0.0  0.0  0.1 16.2  0.1
##          E  0.0  0.0  0.0  0.0 18.3
##
##  Accuracy (average) : 0.9915
```

As we see from the confusion metric and accuracy the result is quite good, let's try to predict on the testing set to test the models.

```
pred_gbm <- predict(fitgbm, testing)
pred_rf <- predict(fitrf, testing)
pred_gbm
```

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

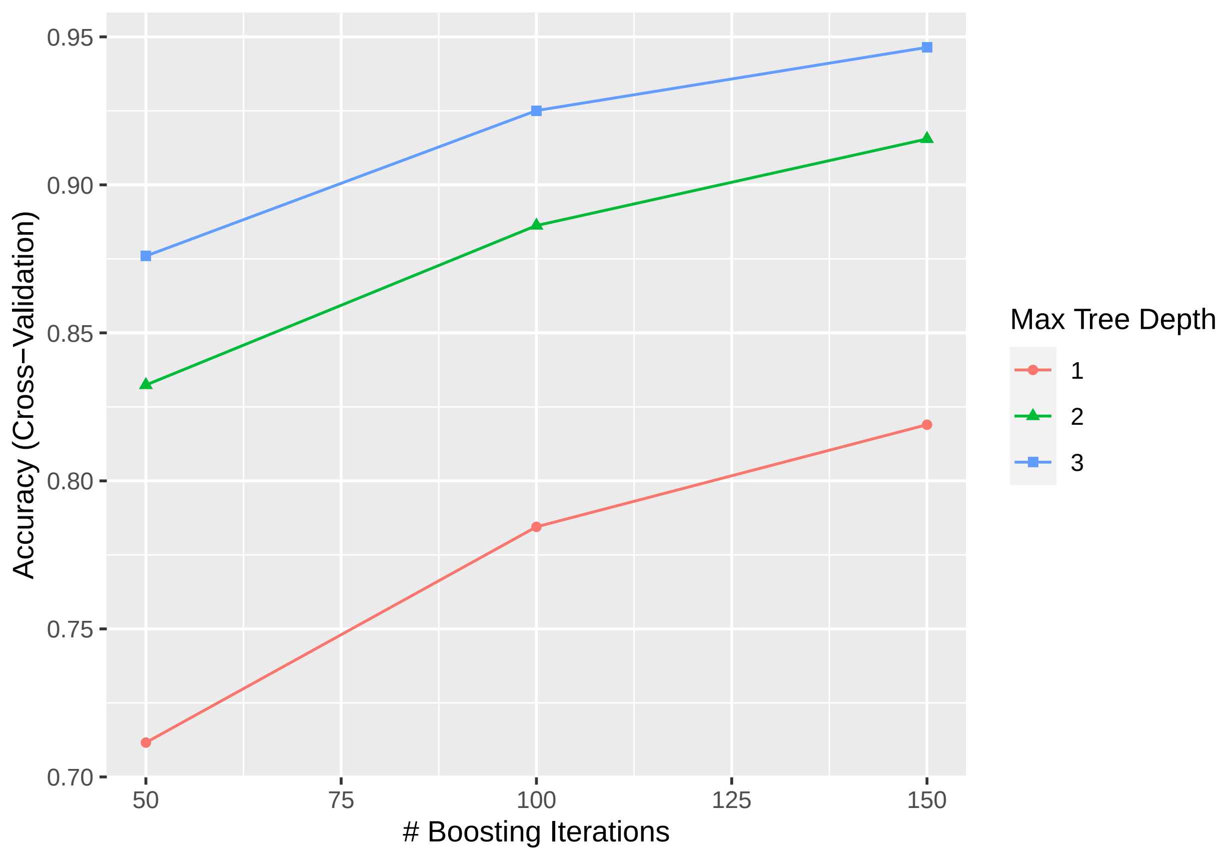
```
pred_rf
```

```
##  [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
```

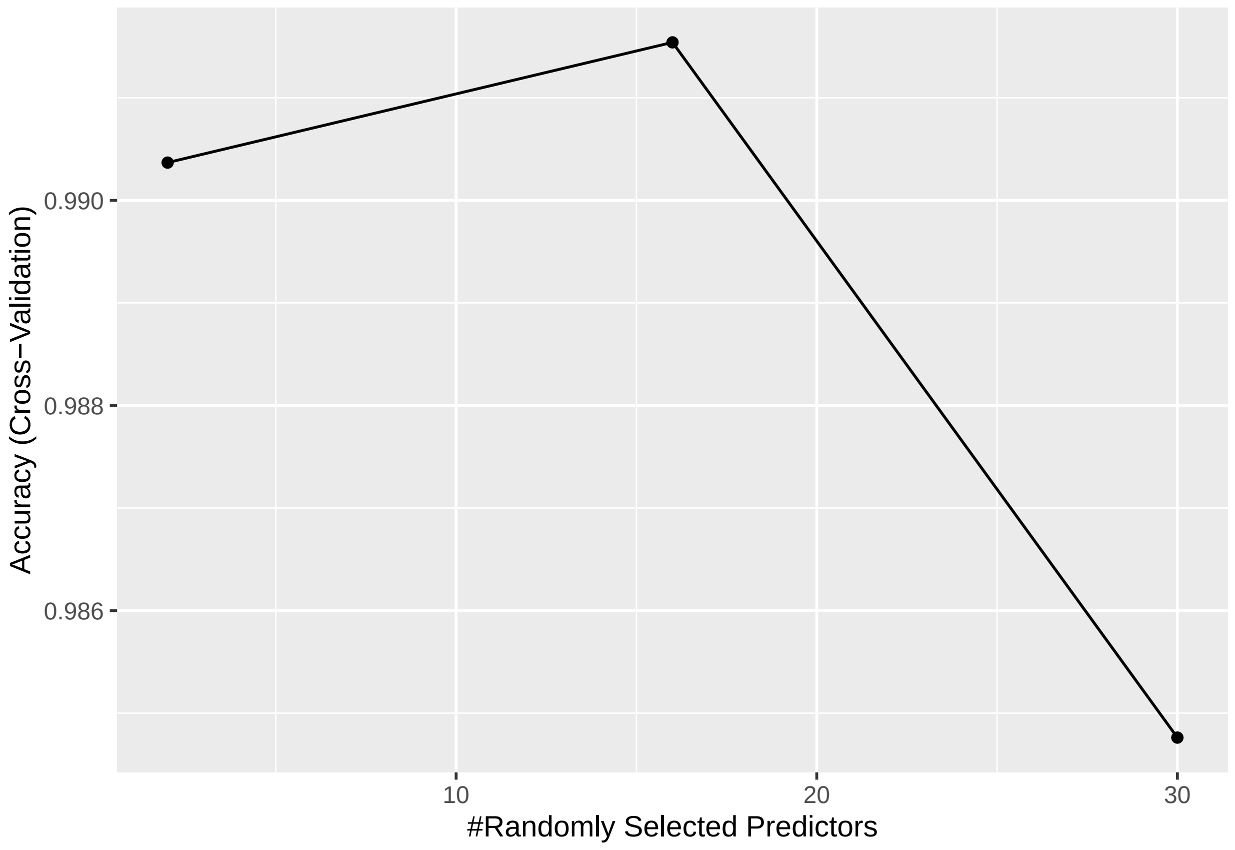
The prediction is the same for both models this is encouraging.

Let's plot the models now to visually see the accuracy

```
ggplot(fitgbm)
```



```
ggplot(fitrf)
```



I think we can be satisfied by the visual result of these plots being the accuracy quite good, both predictions leads to the same results and this is encouraging. I think my work is finished with this, I just couldn't train more models because of the limits of my RAM, but I'm quite confident about the outcome.