Training a Machine Learning algorithm on Human Excercise data

Introduction

For this project, I was asked to train a Machine Learning algorithm to identify the type of excercise being done using data from various wearable sensors.

Exploratory Data Analysis

We begin by loading the data and the necessary libraries and setting a seed to allow for reproducibility:

```
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")
library(caret); library(ggplot2); library(janitor); library(e1071)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
set.seed(24051998)
```

We can see that there are some unnecessary variables , we proceed by cleansing both the training and testing data sets:

```
unn_vars <- c("X","user_name","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestamp","problem_i
training <- training[,-which(names(training) %in% unn_vars)]
testing <- testing[,-which(names(testing) %in% unn_vars)]
training <- training[,-grep("^var|^std|^avg|^kurtosis|^skewness|^amplitude|^max|^min",names(training))]
testing <- testing[,-grep("^var|^std|^avg|^kurtosis|^skewness|^amplitude|^max|^min",names(testing))]
```

We factorize the *new_window* and the *classe* variables and create a validation data set:

```
training$new_window <- as.factor(training$new_window)
training$classe <- as.factor(training$classe)
testing$new_window <- as.factor(testing$new_window)
inTrain <- createDataPartition(training$classe, p = 0.7, list = F)
training <- training[inTrain,]
validation <- training[-inTrain,]
```

The structure of the data:

```
head(training)
```

```
##   new_window num_window roll_belt pitch_belt yaw_belt total_accel_belt
## 1         no         11      1.41       8.07    -94.4                3
## 3         no         11      1.42       8.07    -94.4                3
```

```
## 5         no         12      1.48       8.07    -94.4                3
## 6         no         12      1.45       8.06    -94.4                3
## 7         no         12      1.42       8.09    -94.4                3
## 8         no         12      1.42       8.13    -94.4                3
##   gyros_belt_x gyros_belt_y gyros_belt_z accel_belt_x accel_belt_y accel_belt_z
## 1         0.00         0.00        -0.02          -21            4           22
## 3         0.00         0.00        -0.02          -20            5           23
## 5         0.02         0.02        -0.02          -21            2           24
## 6         0.02         0.00        -0.02          -21            4           21
## 7         0.02         0.00        -0.02          -22            3           21
## 8         0.02         0.00        -0.02          -22            4           21
##   magnet_belt_x magnet_belt_y magnet_belt_z roll_arm pitch_arm yaw_arm
## 1            -3           599          -313     -128      22.5    -161
## 3            -2           600          -305     -128      22.5    -161
## 5            -6           600          -302     -128      22.1    -161
## 6             0           603          -312     -128      22.0    -161
## 7            -4           599          -311     -128      21.9    -161
## 8            -2           603          -313     -128      21.8    -161
##   total_accel_arm gyros_arm_x gyros_arm_y gyros_arm_z accel_arm_x accel_arm_y
## 1              34        0.00        0.00       -0.02        -288         109
## 3              34        0.02       -0.02       -0.02        -289         110
## 5              34        0.00       -0.03        0.00        -289         111
## 6              34        0.02       -0.03        0.00        -289         111
## 7              34        0.00       -0.03        0.00        -289         111
## 8              34        0.02       -0.02        0.00        -289         111
##   accel_arm_z magnet_arm_x magnet_arm_y magnet_arm_z roll_dumbbell
## 1        -123         -368          337          516      13.05217
## 3        -126         -368          344          513      12.85075
## 5        -123         -374          337          506      13.37872
## 6        -122         -369          342          513      13.38246
## 7        -125         -373          336          509      13.12695
## 8        -124         -372          338          510      12.75083
##   pitch_dumbbell yaw_dumbbell total_accel_dumbbell gyros_dumbbell_x
## 1     -70.49400    -84.87394                   37                0
## 3     -70.27812    -85.14078                   37                0
## 5     -70.42856    -84.85306                   37                0
## 6     -70.81759    -84.46500                   37                0
## 7     -70.24757    -85.09961                   37                0
## 8     -70.34768    -85.09708                   37                0
##   gyros_dumbbell_y gyros_dumbbell_z accel_dumbbell_x accel_dumbbell_y
## 1            -0.02                0            -234               47
## 3            -0.02                0            -232               46
## 5            -0.02                0            -233               48
## 6            -0.02                0            -234               48
## 7            -0.02                0            -232               47
## 8            -0.02                0            -234               46
##   accel_dumbbell_z magnet_dumbbell_x magnet_dumbbell_y magnet_dumbbell_z
## 1             -271              -559               293               -65
## 3             -270              -561               298               -63
## 5             -270              -554               292               -68
## 6             -269              -558               294               -66
## 7             -270              -551               295               -70
## 8             -272              -555               300               -74
##   roll_forearm pitch_forearm yaw_forearm total_accel_forearm gyros_forearm_x
```

```
## 1         28.4        -63.9       -153                 36            0.03
## 3         28.3        -63.9       -152                 36            0.03
## 5         28.0        -63.9       -152                 36            0.02
## 6         27.9        -63.9       -152                 36            0.02
## 7         27.9        -63.9       -152                 36            0.02
## 8         27.8        -63.8       -152                 36            0.02
##   gyros_forearm_y gyros_forearm_z accel_forearm_x accel_forearm_y
## 1            0.00           -0.02             192             203
## 3           -0.02            0.00             196             204
## 5            0.00           -0.02             189             206
## 6           -0.02           -0.03             193             203
## 7            0.00           -0.02             195             205
## 8           -0.02            0.00             193             205
##   accel_forearm_z magnet_forearm_x magnet_forearm_y magnet_forearm_z classe
## 1            -215              -17              654              476      A
## 3            -213              -18              658              469      A
## 5            -214              -17              655              473      A
## 6            -215               -9              660              478      A
## 7            -215              -18              659              470      A
## 8            -213               -9              660              474      A
```

```
str(training)
```

```
## 'data.frame':    13737 obs. of  55 variables:
##  $ new_window          : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window          : int  11 11 12 12 12 12 12 12 12 12 ...
##  $ roll_belt           : num  1.41 1.42 1.48 1.45 1.42 1.42 1.45 1.43 1.45 1.48 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.06 8.09 8.13 8.18 8.18 8.2 8.15 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0 0.02 0.02 0.02 0.02 0.03 0.02 0 0 ...
##  $ gyros_belt_y        : num  0 0 0.02 0 0 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 0 0 ...
##  $ accel_belt_x        : int  -21 -20 -21 -21 -22 -22 -21 -22 -21 -21 ...
##  $ accel_belt_y        : int  4 5 2 4 3 4 2 2 2 4 ...
##  $ accel_belt_z        : int  22 23 24 21 21 21 23 23 22 23 ...
##  $ magnet_belt_x       : int  -3 -2 -6 0 -4 -2 -5 -2 -1 0 ...
##  $ magnet_belt_y       : int  599 600 600 603 599 603 596 602 597 592 ...
##  $ magnet_belt_z       : int  -313 -305 -302 -312 -311 -313 -317 -319 -310 -305 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -129 -129 ...
##  $ pitch_arm           : num  22.5 22.5 22.1 22 21.9 21.8 21.5 21.5 21.4 21.3 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0 0.02 0 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 0 0 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 0 0 0 0 0 0 -0.03 -0.03 ...
##  $ accel_arm_x         : int  -288 -289 -289 -289 -289 -289 -290 -288 -289 -289 ...
##  $ accel_arm_y         : int  109 110 111 111 111 111 110 111 111 109 ...
##  $ accel_arm_z         : int  -123 -126 -123 -122 -125 -124 -123 -123 -124 -121 ...
##  $ magnet_arm_x        : int  -368 -368 -374 -369 -373 -372 -366 -363 -374 -367 ...
##  $ magnet_arm_y        : int  337 344 337 342 336 338 339 343 342 340 ...
##  $ magnet_arm_z        : int  516 513 506 513 509 510 509 520 510 509 ...
##  $ roll_dumbbell       : num  13.1 12.9 13.4 13.4 13.1 ...
##  $ pitch_dumbbell      : num  -70.5 -70.3 -70.4 -70.8 -70.2 ...
##  $ yaw_dumbbell        : num  -84.9 -85.1 -84.9 -84.5 -85.1 ...
```

```
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -232 -233 -234 -232 -234 -233 -233 -234 -233 ...
##  $ accel_dumbbell_y    : int  47 46 48 48 47 46 47 47 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -270 -270 -269 -270 -272 -269 -270 -270 -271 ...
##  $ magnet_dumbbell_x   : int  -559 -561 -554 -558 -551 -555 -564 -554 -554 -554 ...
##  $ magnet_dumbbell_y   : int  293 298 292 294 295 300 299 291 294 297 ...
##  $ magnet_dumbbell_z   : num  -65 -63 -68 -66 -70 -74 -64 -65 -63 -73 ...
##  $ roll_forearm        : num  28.4 28.3 28 27.9 27.9 27.8 27.6 27.5 27.2 27.1 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 -63.9 -64 ...
##  $ yaw_forearm         : num  -153 -152 -152 -152 -152 -152 -152 -152 -151 -151 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.03 0.02 0.02 0.02 0.02 0.02 0.02 0 0.02 ...
##  $ gyros_forearm_y     : num  0 -0.02 0 -0.02 0 -0.02 -0.02 0.02 -0.02 0 ...
##  $ gyros_forearm_z     : num  -0.02 0 -0.02 -0.03 -0.02 0 -0.02 -0.03 -0.02 0 ...
##  $ accel_forearm_x     : int  192 196 189 193 195 193 193 191 192 194 ...
##  $ accel_forearm_y     : int  203 204 206 203 205 205 205 203 201 204 ...
##  $ accel_forearm_z     : int  -215 -213 -214 -215 -215 -213 -214 -215 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -17 -9 -18 -9 -17 -11 -16 -13 ...
##  $ magnet_forearm_y    : num  654 658 655 660 659 660 657 657 656 656 ...
##  $ magnet_forearm_z    : num  476 469 473 478 470 474 465 478 472 471 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

The unique values to be predicted are:

```
unique(training$classe)
```

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

As we can see after the cleansing, the training data is comprised of 55 variables and 13737 observations.

Model Fitting

In this project we will try to fit 5 different models and validate/combine them to reach a good enough accuracy. The models we will use are:

- Random Forests
- Linear Discriminant Analysis
- Naive Bayes
- Boosting with Trees
- Support Vector Machine
- Combination of all of them

Random Forests

We fit the model:

```
model_rf <- train(classe ~ ., model = "rf", data = training)
model_rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, model = "rf")
##                Type of random forest: classification
```

```
##                       Number of trees: 500
## No. of variables tried at each split: 28
##
##         OOB estimate of  error rate: 0.25%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    8 2643    6    1    0 0.0056433409
## C    0    3 2393    0    0 0.0012520868
## D    0    0    9 2242    1 0.0044404973
## E    0    1    0    4 2520 0.0019801980
```

Then predict on the testing data set:

```
predict_rf <- predict(model_rf, validation)
head(predict_rf)
```

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

The accuracy for this model is:

```
confusionMatrix(predict_rf, validation$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9991055      1.0000000      0.2850558
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Linear Dsicriminant Analysis

We begin by fitting the model

```
model_lda <- train(classe ~ ., model = "lda", data = training)
model_lda$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, model = "lda")
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 28
##
##         OOB estimate of  error rate: 0.25%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    9 2645    3    1    0 0.0048908954
## C    0    3 2392    1    0 0.0016694491
## D    0    0   10 2241    1 0.0048845471
## E    0    1    0    3 2521 0.0015841584
```

Then predict on the testing data set:

```
predict_lda <- predict(model_lda, validation)
head(predict_lda)
```

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

The accuracy for this model is:

```
confusionMatrix(predict_lda, validation$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9991055      1.0000000      0.2850558
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Naive Bayes

We begin by fitting the model

```
model_nb <- train(classe ~ ., model = "nb", data = training)
model_nb$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, model = "nb")
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 28
##
##         OOB estimate of  error rate: 0.25%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B   11 2642    4    1    0 0.0060195636
## C    0    3 2393    0    0 0.0012520868
## D    0    0    7 2244    1 0.0035523979
## E    0    1    0    4 2520 0.0019801980
```

Then predict on the testing data set:

```
predict_nb <- predict(model_nb, validation)
head(predict_nb)
```

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

The accuracy for this model is:

```
confusionMatrix(predict_nb, validation$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9991055      1.0000000      0.2850558
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Boosting with Trees

We begin by fitting the model

```
model_gbm <- train(classe ~ ., model = "gbm", data = training)
model_gbm$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, model = "gbm")
```

```
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 28
##
##         OOB estimate of  error rate: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    8 2647    2    1    0 0.0041384500
## C    0    3 2393    0    0 0.0012520868
## D    0    0    7 2244    1 0.0035523979
## E    0    1    0    3 2521 0.0015841584
```

Then predict on the testing data set:

```
predict_gbm <- predict(model_gbm, validation)
head(predict_gbm)
```

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

The accuracy for this model is:

```
confusionMatrix(predict_gbm, validation$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9991055      1.0000000      0.2850558
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Support Vector Machine

We begin by fitting the model

```
model_svm <- svm(classe ~ ., data = training)
model_svm
```

```
##
## Call:
## svm(formula = classe ~ ., data = training)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  6322
```

Then predict on the testing data set:

```
predict_svm <- predict(model_svm, validation)
head(predict_svm)
```

```
##  3  6 15 16 21 22
##  A  A  A  A  A  A
## Levels: A B C D E
```

The accuracy for this model is:

```
confusionMatrix(predict_svm, validation$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9524503      0.9397458      0.9455043      0.9587455      0.2850558
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Combination

Since we now have all four models, we can use a combination of all using Random Forests once again to get a better accuracy:

```
df_all <- data.frame(predict_rf, predict_lda, predict_nb, predict_gbm, predict_svm, classe = validation$
model_all <- train(classe ~ ., method = "rf", data = df_all)
model_all$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0%
## Confusion matrix:
##      A   B   C   D   E class.error
## A 1175   0   0   0   0           0
## B    0 789   0   0   0           0
## C    0   0 739   0   0           0
## D    0   0   0 642   0           0
## E    0   0   0   0 777           0
```

Now we try to predict once again using this new model:

```
predict_all <- predict(model_all, validation)
head(predict_all)
```

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

With this accuracy:

```
confusionMatrix(predict_all, validation$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9991055      1.0000000      0.2850558
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Predictions on Test data set

Since all models have a high accuracy, we will proceed to use only the Random Forest model to try to predict the results on the test set, which will be used on the quiz:

```
predict_test <- predict(model_rf,testing)
predict_test
```

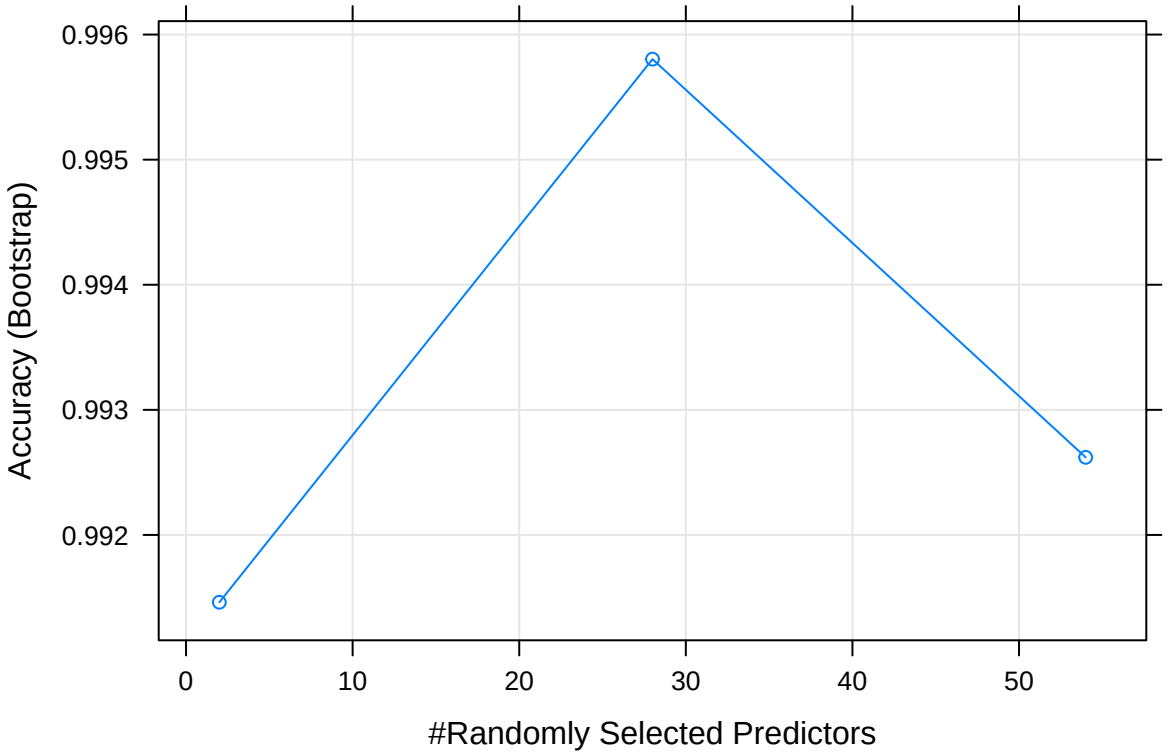
```
##  [1] B A B A A E D B A A B C B A E E A B B B
```

```
## Levels: A B C D E
```

Appendix

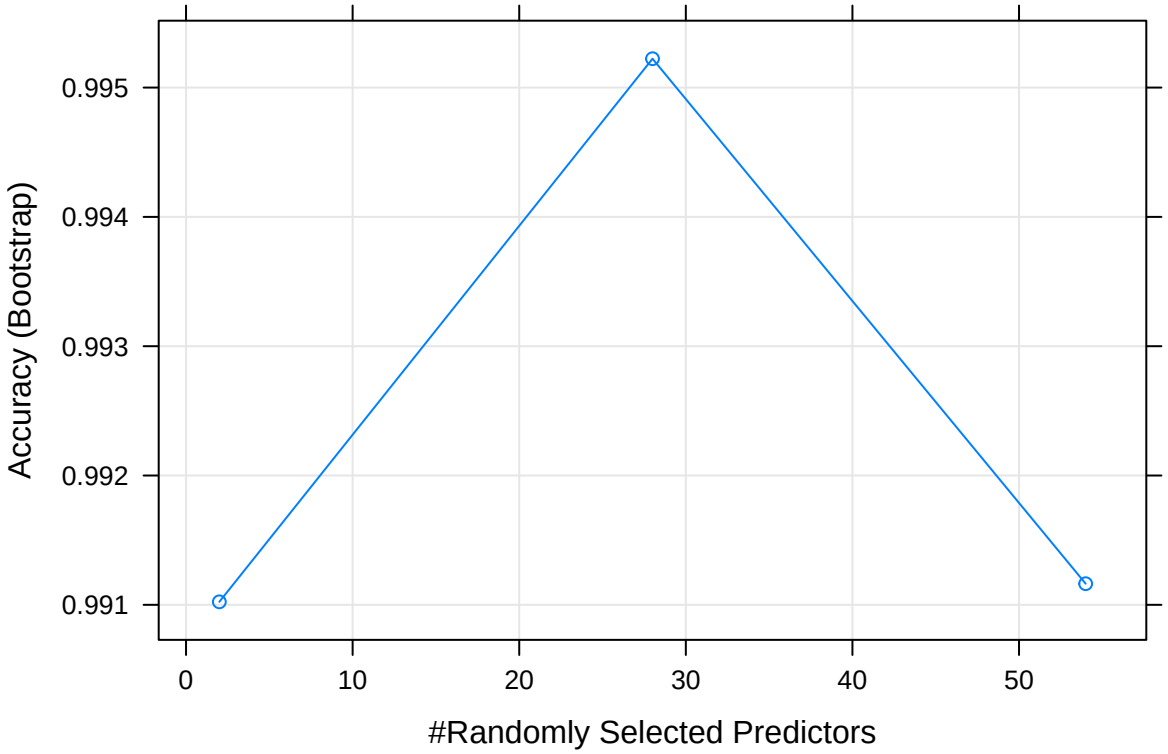
Plot of the Random Forest model:

```
plot(model_rf)
```



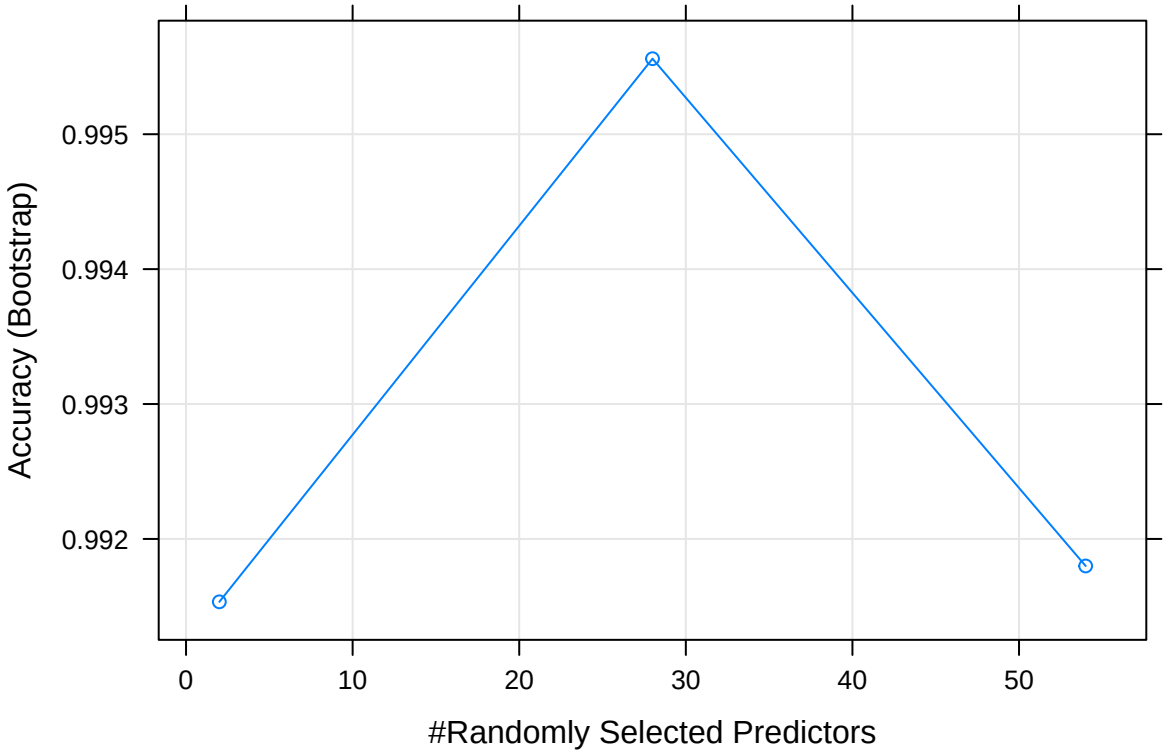
Plot of the General Boosted Trees model:

```
plot(model_gbm)
```



Plot of the Linear Discriminant Analysis model:

```
plot(model_lda)
```



Plot of the Naive Bayes model:

```
plot(model_nb)
```