Practical Machine Learning Project

Ashwin Sai Murali Neelakandan

11/11/2024

Introduction

The goal of this project is to use data from accelerometers on the belt, forearm, arm, and dumbbells of 6 participants who were asked to perform barbell lifts correctly and incorrectly in 5 different ways, to predict the manner in which the participants did the exercise.

Data

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.

The training data for this project are available here:

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

The test data are available here:

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har

Loading the necessary packages

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2
```

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

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

```r
library(lattice)
library(ggplot2)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

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

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.4.2
```

```
## randomForest 4.7-1.2
```

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

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

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

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.4.2
```

```
## corrplot 0.95 loaded
```

```r
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.4.2
```

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

Loading and Data Preprocessing

```r
raw_train <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
raw_test <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
# Looking at the dimensions of the training and test data
dim(raw_train)
```

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

```r
dim(raw_test)
```

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

Seeing if the data has any NA values and then replacing most of them with 0

```r
# Seeing which observations don't have missing (NA) values
sum(complete.cases(raw_train))
```

```
## [1] 406
```

Selecting and replacing the missing values in both training and test sets

```r
raw_train <- raw_train[, colMeans(is.na(raw_train)) < .9]
test_data <- raw_test[, colMeans(is.na(raw_test))<.9]
# Looking at the names and nature of the column variables
str(raw_train)
```

```
## 'data.frame':    19622 obs. of  93 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 13
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 48443
##  $ cvtd_timestamp          : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "05/12/201
##  $ new_window              : chr  "no" "no" "no" "no" ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ 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 ...
##  $ kurtosis_roll_belt      : chr  "" "" "" "" ...
##  $ kurtosis_picth_belt     : chr  "" "" "" "" ...
##  $ kurtosis_yaw_belt       : chr  "" "" "" "" ...
##  $ skewness_roll_belt      : chr  "" "" "" "" ...
##  $ skewness_roll_belt.1    : chr  "" "" "" "" ...
##  $ skewness_yaw_belt       : chr  "" "" "" "" ...
##  $ max_yaw_belt            : chr  "" "" "" "" ...
##  $ min_yaw_belt            : chr  "" "" "" "" ...
##  $ amplitude_yaw_belt      : chr  "" "" "" "" ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ 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 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
```

```
##  $ magnet_arm_x           : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y           : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z           : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm      : chr  "" "" "" "" ...
##  $ kurtosis_picth_arm     : chr  "" "" "" "" ...
##  $ kurtosis_yaw_arm       : chr  "" "" "" "" ...
##  $ skewness_roll_arm      : chr  "" "" "" "" ...
##  $ skewness_pitch_arm     : chr  "" "" "" "" ...
##  $ skewness_yaw_arm       : chr  "" "" "" "" ...
##  $ roll_dumbbell          : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell         : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell           : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell : chr  "" "" "" "" ...
##  $ kurtosis_picth_dumbbell: chr  "" "" "" "" ...
##  $ kurtosis_yaw_dumbbell  : chr  "" "" "" "" ...
##  $ skewness_roll_dumbbell : chr  "" "" "" "" ...
##  $ skewness_pitch_dumbbell: chr  "" "" "" "" ...
##  $ skewness_yaw_dumbbell  : chr  "" "" "" "" ...
##  $ max_yaw_dumbbell       : chr  "" "" "" "" ...
##  $ min_yaw_dumbbell       : chr  "" "" "" "" ...
##  $ amplitude_yaw_dumbbell : chr  "" "" "" "" ...
##  $ 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 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x       : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y       : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z       : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x      : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y      : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z      : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm           : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm          : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm            : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ kurtosis_roll_forearm  : chr  "" "" "" "" ...
##  $ kurtosis_picth_forearm : chr  "" "" "" "" ...
##  $ kurtosis_yaw_forearm   : chr  "" "" "" "" ...
##  $ skewness_roll_forearm  : chr  "" "" "" "" ...
##  $ skewness_pitch_forearm : chr  "" "" "" "" ...
##  $ skewness_yaw_forearm   : chr  "" "" "" "" ...
##  $ max_yaw_forearm        : chr  "" "" "" "" ...
##  $ min_yaw_forearm        : chr  "" "" "" "" ...
##  $ amplitude_yaw_forearm  : chr  "" "" "" "" ...
##  $ total_accel_forearm    : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x        : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y        : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z        : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x        : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y        : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z        : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x       : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y       : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z       : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe                 : chr  "A" "A" "A" "A" ...
```

```
# Removing the columns with data (S.No, name, timestamp, and window) which are irrelevant to the outcom
raw_train <- raw_train[, -c(1:7)]
```

Removing near zero variance variables from the raw training set

```
# For training set
nearzvar1 <- nearZeroVar(raw_train)
raw_train <- raw_train[, -nearzvar1]
# Looking at the dimensions of training data after pre-processing
dim(raw_train)
```

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

Splitting the Training dataset

```
# Setting a seed for reproducibility
set.seed(6583)
# Partitioning  the cleaned dataset into training and validation data sets
inData <- createDataPartition(y=raw_train$classe, p=0.7, list = FALSE)
Traindata <- raw_train[inData,]
Validdata <- raw_train[-inData,]
```

The cleaned data was split into a training set (70%) and a validation set (30%) which will be used for cross validation purposes.

```
# Converting the classe variable into a factor variable
Traindata$classe <- as.factor(Traindata$classe)
Validdata$classe <- as.factor(Validdata$classe)
# Setting up a control to use 5-fold cross validation for Decision Trees, Random Forests, and Support V
crossvalid_control <- trainControl(method="cv", number=5, verboseIter=FALSE)
```

Model Building

Trying fit the prediction model based on a few popular model approaches:
i. Decision Trees
ii. Random Forests
iii. Support Vector Machines
iv. Generalized Boosting

Decision Trees

```
# Creating a Decision tree prediction model using the rpart method
Tree_mod <- train(classe ~ ., data = Traindata, method = "rpart",
                  trControl = crossvalid_control)
# Applying the model to the validation set
Tree_pred <- predict(Tree_mod, Validdata)
Tree_cfm <- confusionMatrix(Tree_pred, Validdata$classe)
Tree_cfm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1511  459  484  449  148
##          B   18  296   15  161   66
##          C  141  384  527  354  387
##          D    0    0    0    0    0
##          E    4    0    0    0  481
##
## Overall Statistics
##
##                Accuracy : 0.4783
##                  95% CI : (0.4655, 0.4912)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.319
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9026  0.25988  0.51365   0.0000  0.44455
## Specificity            0.6343  0.94522  0.73945   1.0000  0.99917
## Pos Pred Value         0.4952  0.53237  0.29392      NaN  0.99175
## Neg Pred Value         0.9425  0.84181  0.87805   0.8362  0.88870
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2568  0.05030  0.08955   0.0000  0.08173
## Detection Prevalence   0.5184  0.09448  0.30467   0.0000  0.08241
## Balanced Accuracy      0.7685  0.60255  0.62655   0.5000  0.72186
```

```r
Tree_accuracy <- Tree_cfm$overall[1]
Tr_outsamperror <- 1 - Tree_accuracy
```

The accuracy obtained for the Decision trees model is 0.4783347 and the out of sample error rate is 0.5216653

Random Forests

```r
# Creating a Random Forests prediction model with 5-fold cross validation using the rf method
RF_mod <- train(classe~., Traindata, method = "rf",
                trControl = crossvalid_control)
# Applying the model to the validation set
RF_pred <- predict(RF_mod, Validdata)
RF_cfm <- confusionMatrix(RF_pred, Validdata$classe)
RF_cfm
```

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

```
## Prediction    A    B    C    D    E
##          A 1671    2    0    0    0
##          B    2 1135   11    0    0
##          C    1    2 1013   18    0
##          D    0    0    2  946    2
##          E    0    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.9932
##                  95% CI : (0.9908, 0.9951)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9914
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9965   0.9873   0.9813   0.9982
## Specificity            0.9995   0.9973   0.9957   0.9992   1.0000
## Pos Pred Value         0.9988   0.9887   0.9797   0.9958   1.0000
## Neg Pred Value         0.9993   0.9992   0.9973   0.9964   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1929   0.1721   0.1607   0.1835
## Detection Prevalence   0.2843   0.1951   0.1757   0.1614   0.1835
## Balanced Accuracy      0.9989   0.9969   0.9915   0.9903   0.9991
```

```
RF_accuracy <- RF_cfm$overall[1]
RF_outsamperror <- 1 - RF_accuracy
```

The accuracy obtained for the Random Forests model is 0.9932031 and the out of sample error rate is 0.0067969

Support Vector Machine

```
# Creating a Support Vector Machine prediction model with 5-fold cross validation using the svmLinear m
SVM_mod <- train(classe~., Traindata, method = "svmLinear",
                 trControl = crossvalid_control)
# Applying the model to the validation set
SVM_pred <- predict(SVM_mod, Validdata)
SVM_cfm <- confusionMatrix(SVM_pred, Validdata$classe)
SVM_cfm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1553  152   80   66   55
```

```
##          B   26  811   80   38  134
##          C   48   74  811  107   74
##          D   42   24   31  702   59
##          E    5   78   24   51  760
##
## Overall Statistics
##
##                Accuracy : 0.7879
##                  95% CI : (0.7773, 0.7983)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7304
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9277   0.7120   0.7904   0.7282   0.7024
## Specificity            0.9162   0.9414   0.9376   0.9683   0.9671
## Pos Pred Value         0.8148   0.7447   0.7280   0.8182   0.8279
## Neg Pred Value         0.9696   0.9316   0.9549   0.9479   0.9352
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2639   0.1378   0.1378   0.1193   0.1291
## Detection Prevalence   0.3239   0.1850   0.1893   0.1458   0.1560
## Balanced Accuracy      0.9219   0.8267   0.8640   0.8483   0.8348
```

```
SVM_accuracy <- SVM_cfm$overall[1]
SVM_outsamperror <- 1 - SVM_accuracy
```

The accuracy obtained for the Support Vector Machine model is 0.7879354 and the out of sample error rate is 0.2120646

Generalized Boosting

```
# Setting up a separate control to use repeated 5-fold cross validation for the Generalized Boosting mo
Gbm_Control <- trainControl(method = "repeatedcv", number = 5, verboseIter = FALSE)
# Creating a Generalized Boosting prediction model with 5-fold repeated cross validation using the gbm
Gbm_mod <- train(classe~., Traindata, method = "gbm",
                 trControl = Gbm_Control,
                 verbose = FALSE)
# Applying the model to the validation set
Gbm_pred <- predict(Gbm_mod, Validdata)
Gbm_cfm <- confusionMatrix(Gbm_pred, Validdata$classe)
Gbm_cfm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1641   27    0    1    1
##          B   18 1071   30    4   13
##          C   10   41  983   41   12
##          D    5    0   12  908   13
##          E    0    0    1   10 1043
##
## Overall Statistics
##
##                Accuracy : 0.9594
##                  95% CI : (0.954, 0.9643)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9486
##
##  Mcnemar's Test P-Value : 4.063e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9803   0.9403   0.9581   0.9419   0.9640
## Specificity            0.9931   0.9863   0.9786   0.9939   0.9977
## Pos Pred Value         0.9826   0.9428   0.9043   0.9680   0.9896
## Neg Pred Value         0.9922   0.9857   0.9910   0.9887   0.9919
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2788   0.1820   0.1670   0.1543   0.1772
## Detection Prevalence   0.2838   0.1930   0.1847   0.1594   0.1791
## Balanced Accuracy      0.9867   0.9633   0.9683   0.9679   0.9808
```

```r
Gbm_accuracy <- Gbm_cfm$overall[1]
Gbm_outsamperror <- 1 - Gbm_accuracy
```

The accuracy obtained for the Generalized Boosting model is 0.9593883 and the out of sample error rate is 0.0406117

Selecting the Prediction model based on Accuracy and Out of Sample Error rate

Creating a table with Accuracy and out of sample error rates for all the above models

```r
model_names = c("Tree", "RandomForests", "SupportVectorMachine", "GeneralizedBoosting")
Accuracy <- round(c(Tree_accuracy, RF_accuracy, SVM_accuracy, Gbm_accuracy), 4)
Out_of_Sample_Error <- 1-Accuracy
data.frame(Accuracy=Accuracy, Out_of_Sample_Error_Rate = Out_of_Sample_Error, row.names = model_names)
```

```
##                      Accuracy Out_of_Sample_Error_Rate
## Tree                   0.4783                   0.5217
## RandomForests          0.9932                   0.0068
## SupportVectorMachine   0.7879                   0.2121
## GeneralizedBoosting    0.9594                   0.0406
```

Based on the results observed, the Random Forests model has the highest accuracy of 0.9932031 and the lowest out of sample error rate of 0.0067969 . The Generalized Boosting model has the second highest accuracy of 0.9593883 and out of sample error rate of 0.0406117.

Therefore, the Random Forests model is selected as the optimal prediction model.

Applying the Random Forests Prediction model to the Test data set

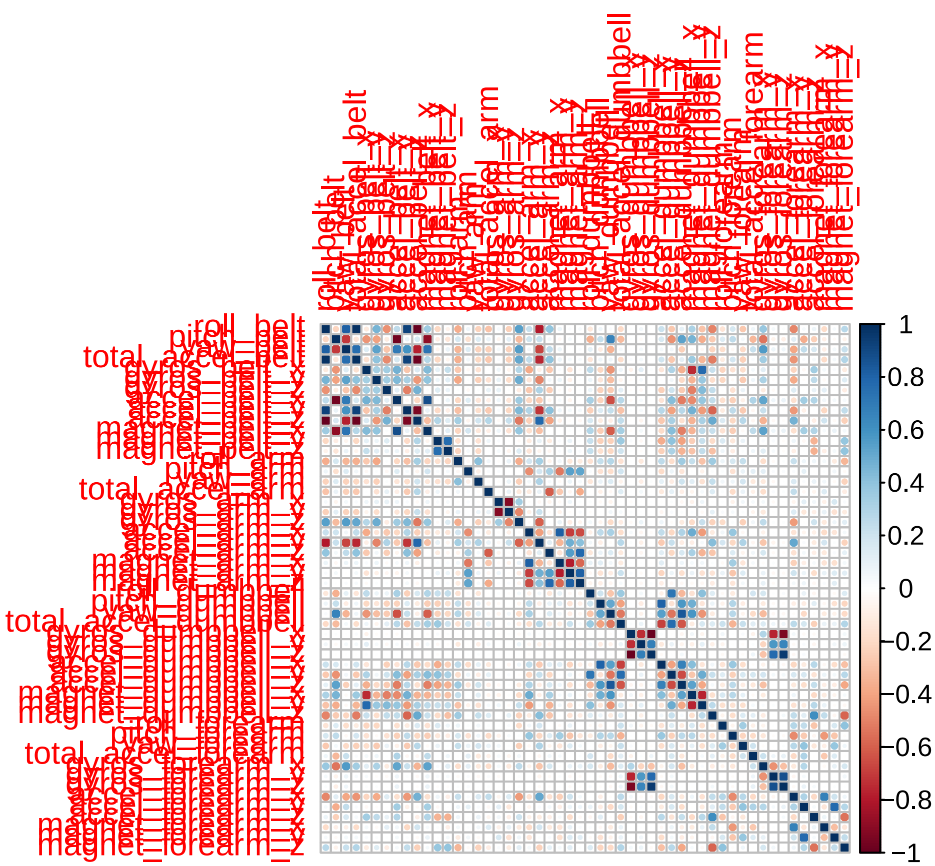
```
Pred_Results <- predict(RF_mod, test_data)
Pred_Results
```

```
##  [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: Plots

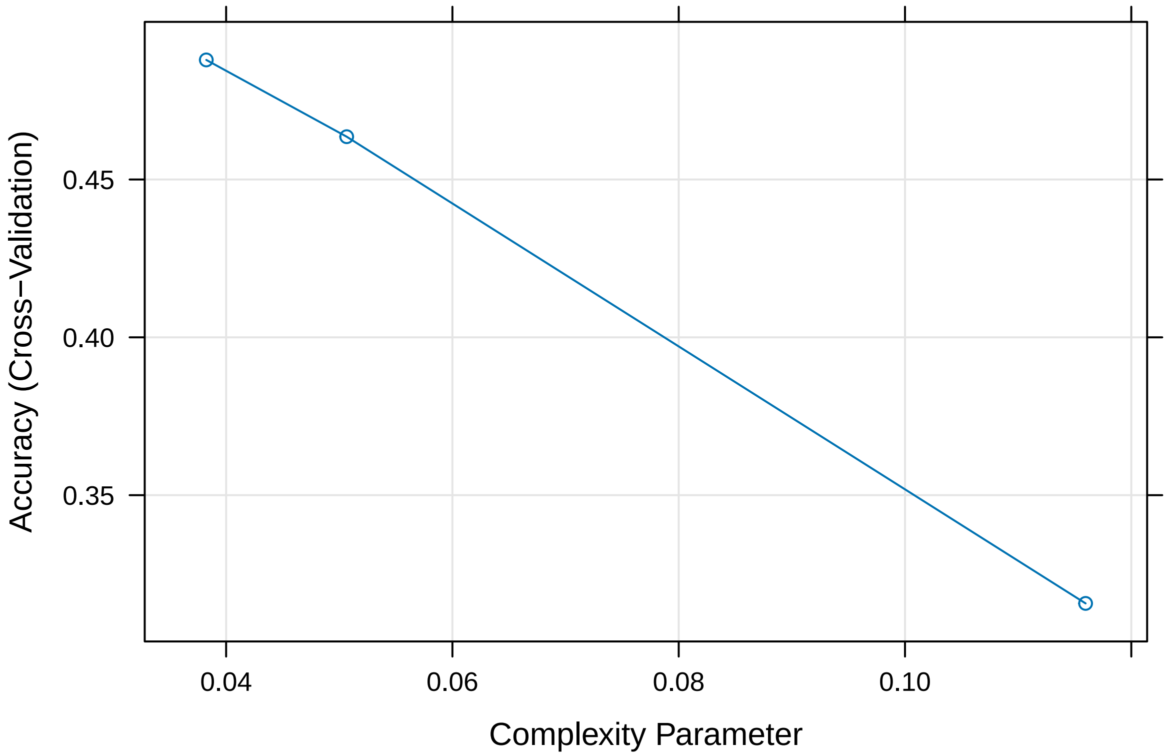
Correlation Plot of variables in the training set

```
corr_matrix <- cor(Traindata[, -length(names(Traindata))])
corrplot(corr_matrix, method = "circle")
```

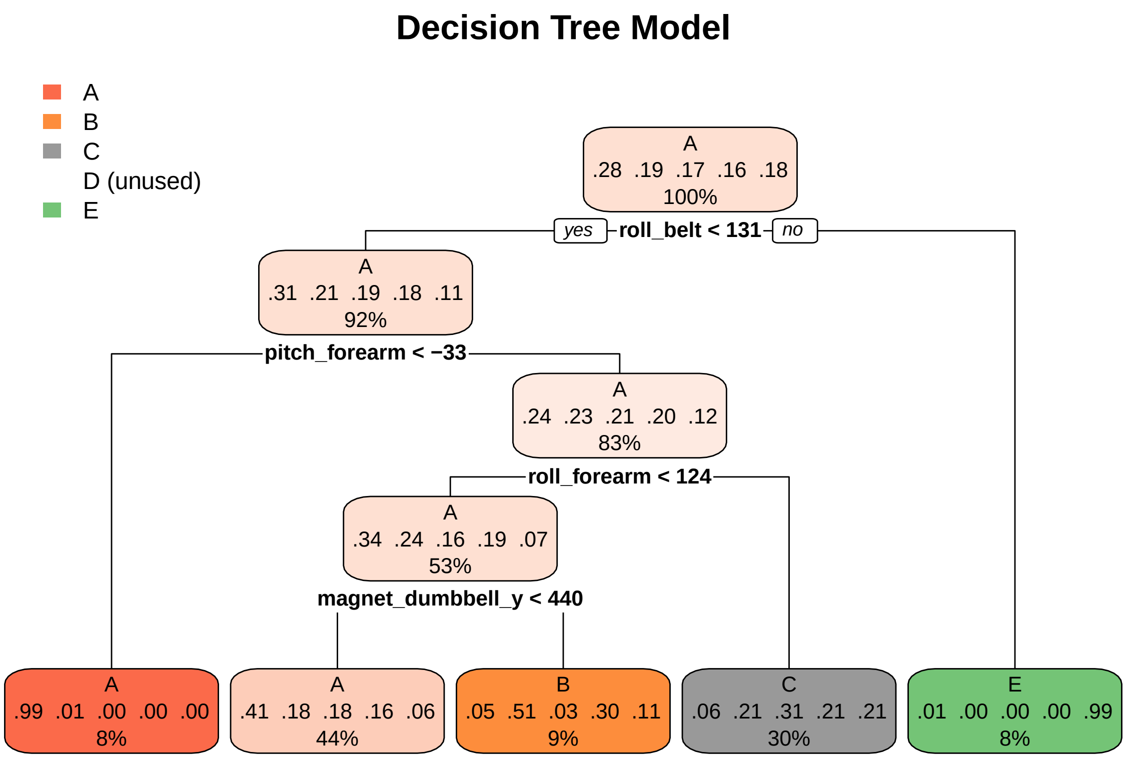


Plotting the different models

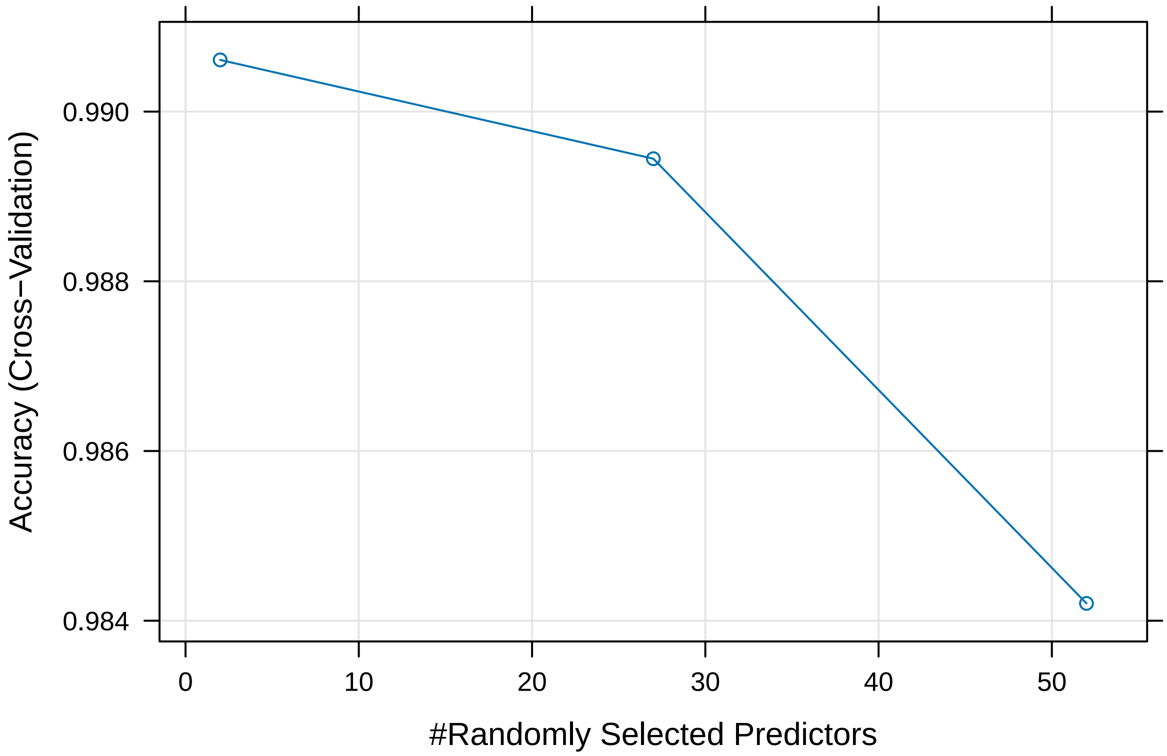
```
# Plotting Decision Tree model
plot(Tree_mod)
```



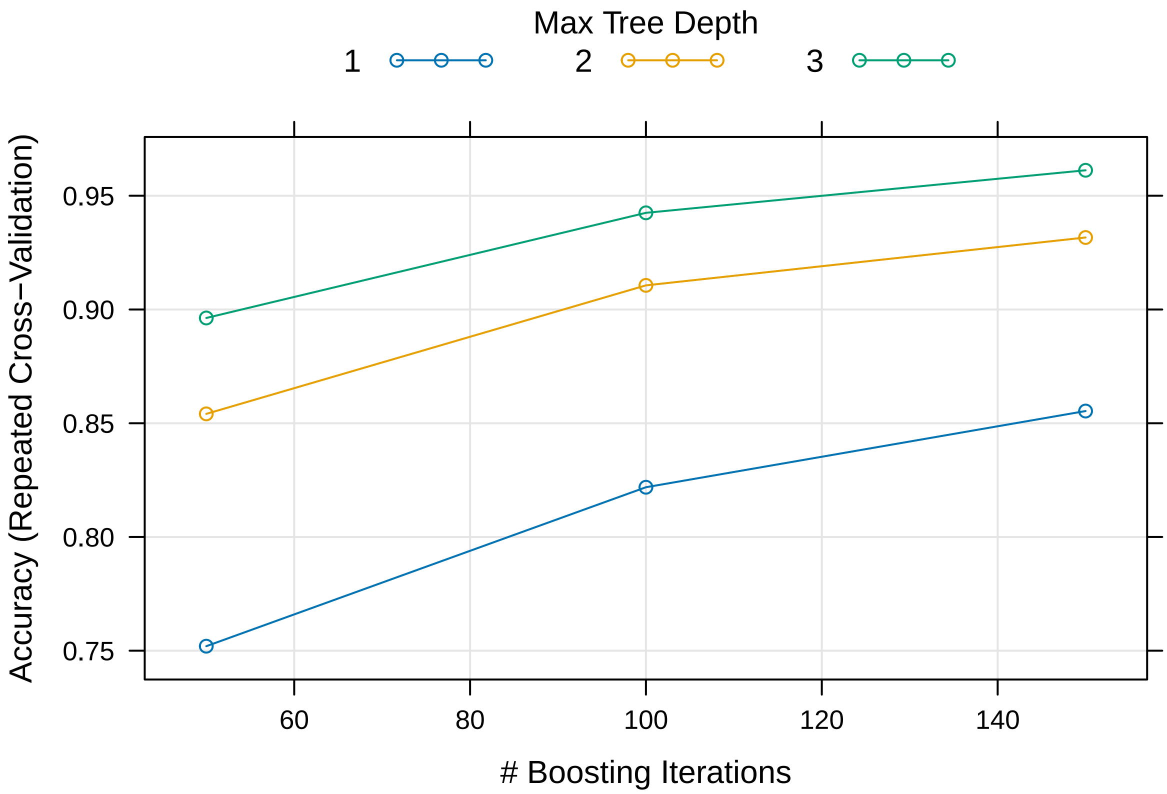

```r
# Visualizing the Decision Tree model
rpart.plot(Tree_mod$finalModel, main = "Decision Tree Model")
```




```r
# Plotting Random Forests model
plot(RF_mod)
```




```r
# Plotting the Generalized Boosting Model
plot(Gbm_mod)
```


Max Tree Depth
1
2
3
Accuracy (Repeated Cross−Validation)
0.95
0.90
0.85
0.80
0.75
60
80
100
120
140
Boosting Iterations