Practical Machine Learning

Pin

2024-11-16

R Markdown

This report aims to evaluate the performance of four different machine learning models—Decision Tree, Random Forests, Support Vector Machine (SVM), and Generalized Boosting—on predicting the class of a weight lifting exercise dataset. The dataset contains multiple features derived from accelerometers placed on different body parts of six participants performing barbell lifts. Each model is assessed based on its accuracy and out-of-sample error, using a training dataset and a separate validation set. The goal is to determine which model provides the best predictive performance for classifying exercise types based on the sensor data.

Load all necessary packages

```
library(caret)
library(lattice)
library(ggplot2)
library(kernlab)
library(randomForest)
library(corrplot)
library(rpart.plot)
```

Data Loading and Preprocessing

```
train_raw <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
test_raw <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
dim(train_raw)
```

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

```
dim(test_raw)
```

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

Handle the Missing Values

```
# Check the observations that have no missing values
sum(complete.cases(train_raw))
```

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

Replace the missing values in training and testing datasets

```
# Replace the missing values and remove na
train_raw <- train_raw[, colMeans(is.na(train_raw)) < .9]
test_data <- test_raw[, colMeans(is.na(test_raw)) < .9]

# Check the data structure
str(train_raw)
```

```
## '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" ...
```

```r
# Remove column with data
train_raw <- train_raw[, -c(1:7)]
```

Remove if the variances is too close to 0

```r
nearzvar <- nearZeroVar(train_raw)
train_raw <- train_raw[, -nearzvar]

# confirm
dim(train_raw)
```

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

Split the Training Dataset

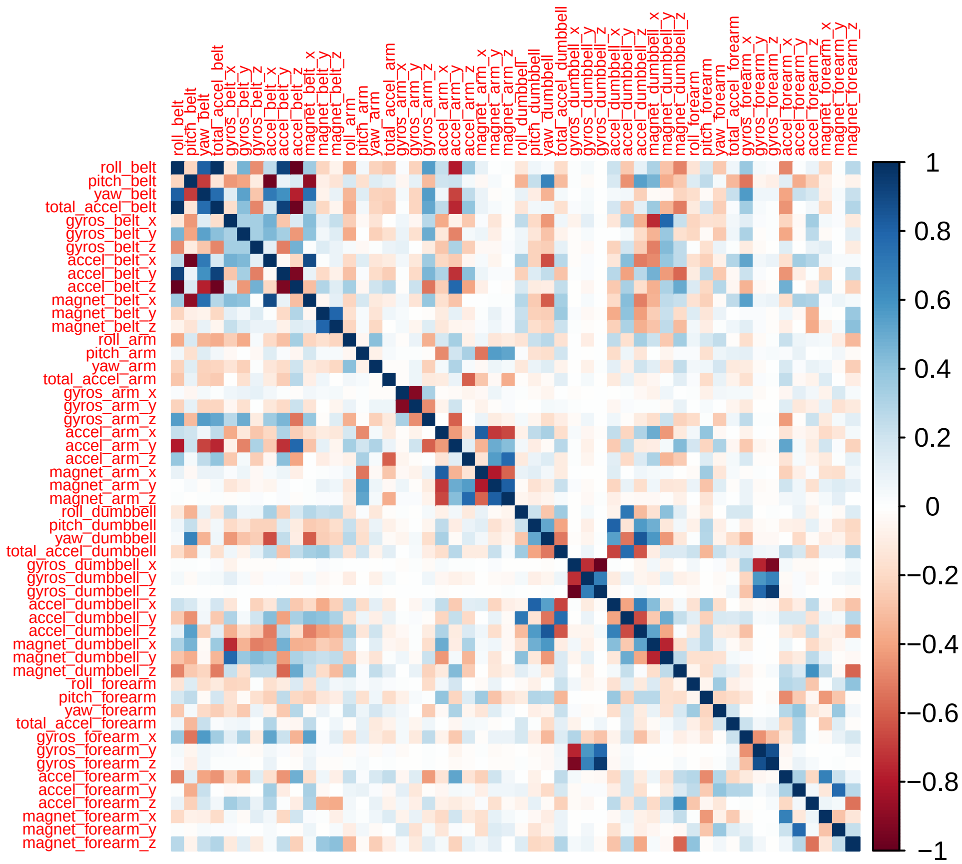
```r
# set seed for reproducibility
set.seed(123)

# partitioning the dataset into training and validation
data <- createDataPartition(y=train_raw$classe, p=0.7, list=FALSE)
train_data <- train_raw[data,]
valid_data <- train_raw[-data,]
```

For cross validation, 70% of the dataset will split into training data (because of p = 0.7) and validation will be 30% of the datasets.

#Plot

```r
# plot
corr_matrix <- cor(train_data[, -length(names(train_data))])
corrplot(corr_matrix, method = "color", tl.cex = 0.5)
```



Convert "classe" into factor variable

```
train_data$classe <- as.factor(train_data$classe)
valid_data$classe <- as.factor(valid_data$classe)
```

Set up control to use 5 fold cross validation for prediction model

```
valid_control <- trainControl(method="cv", number = 5, verboseIter = FALSE)
```

Prediction Model

Application on 4 prediction models: (1) Decision Trees (2) Random Forests (3) Support Vector Machines (4) Generalized Boosting

Decision Trees

```r
# Create a decision tree model by using the rpart method

decision_tree <- train(classe ~ ., data = train_data, method = "rpart",
                       trControl = valid_control)

# Apply the model to the validation set

tree_predict <- predict(decision_tree, valid_data)
tree_confm <- confusionMatrix(tree_predict, valid_data$classe)
tree_confm
```

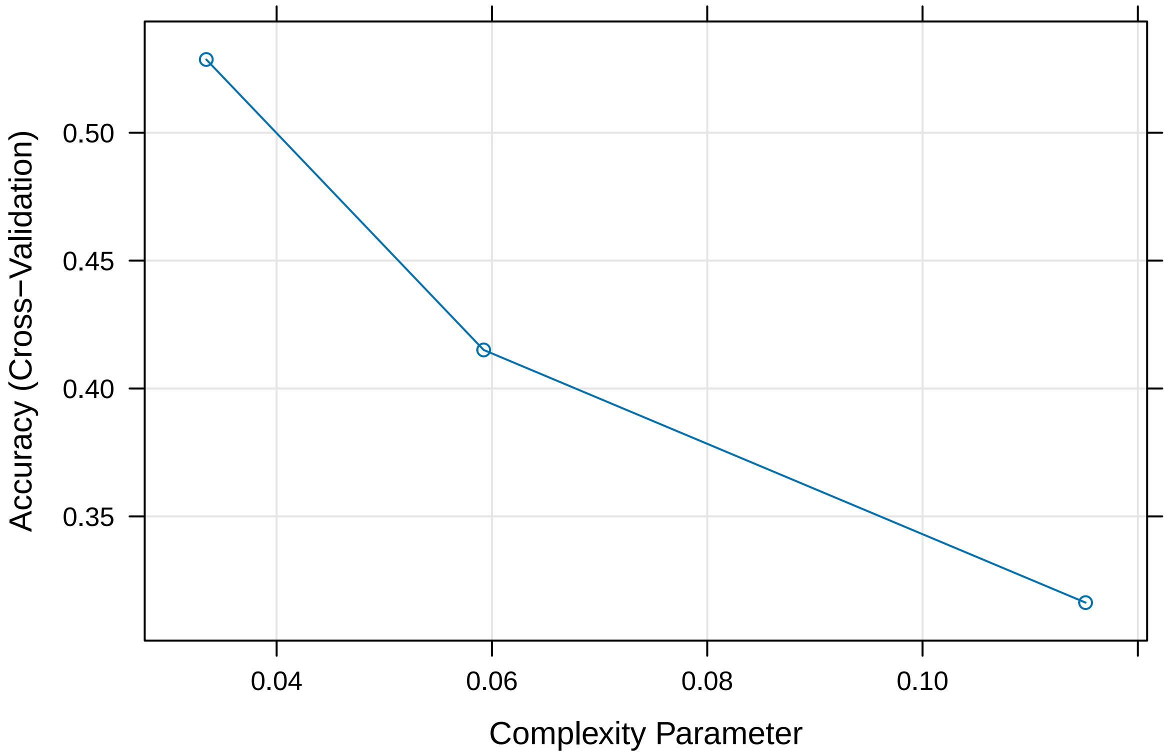
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  464  469  440  144
##          B   28  397   30  169  145
##          C  114  278  527  355  306
##          D    0    0    0    0    0
##          E    2    0    0    0  487
##
## Overall Statistics
##
##                Accuracy : 0.4997
##                  95% CI : (0.4869, 0.5126)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3464
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140  0.34855  0.51365   0.0000  0.45009
## Specificity            0.6398  0.92162  0.78329   1.0000  0.99958
## Pos Pred Value         0.5021  0.51625  0.33354      NaN  0.99591
## Neg Pred Value         0.9493  0.85496  0.88409   0.8362  0.88973
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2600  0.06746  0.08955   0.0000  0.08275
## Detection Prevalence   0.5178  0.13067  0.26848   0.0000  0.08309
## Balanced Accuracy      0.7769  0.63508  0.64847   0.5000  0.72484
```

```r
tree_accuracy <- tree_confm$overall[1]
tree_samper <- 1 - tree_accuracy
tree_samper
```

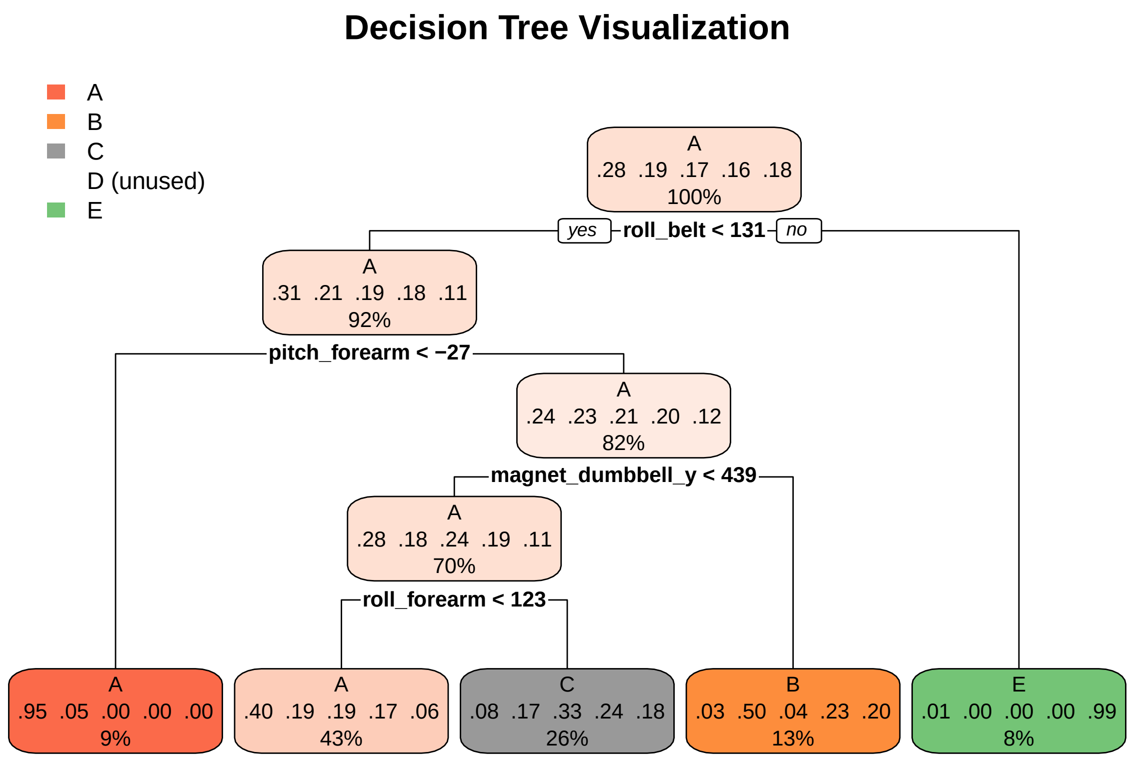
```
##  Accuracy
## 0.5002549
```

The decision tree prediction model show 0.4997 accuracy and the out of sample error rate of 0.5002549.

```r
plot(decision_tree)
```

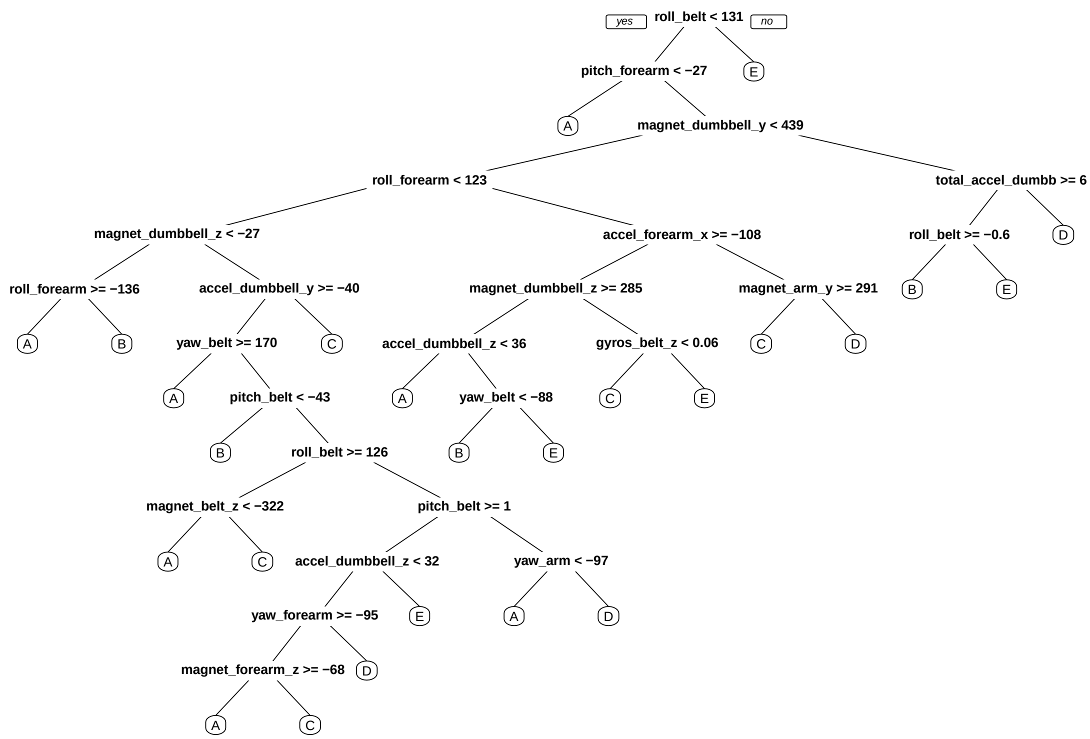



```r
rpart.plot(decision_tree$finalModel, main = "Decision Tree Visualization")
```




```r
# Create rpart decision tree model
model_tree <- rpart(classe ~., data = train_data, method = "class")

# plot decision tree using rpart
prp(model_tree)
```



Random Forest

```r
# Create random forest model with 5 fold cross validation using rf method

rf_model <- train(classe ~ ., data = train_data, method = "rf",
                  trControl = valid_control)

# Apply the model to the validation set
rf_pred <- predict(rf_model, valid_data)
rf_confm <- confusionMatrix(rf_pred, valid_data$classe)
rf_confm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    6    0    0    0
##          B    1 1125    5    0    0
##          C    0    8 1018   11    4
##          D    0    0    3  953    5
##          E    0    0    0    0 1073
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9927
##                  95% CI : (0.9902, 0.9947)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9908
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9877   0.9922   0.9886   0.9917
## Specificity            0.9986   0.9987   0.9953   0.9984   1.0000
## Pos Pred Value         0.9964   0.9947   0.9779   0.9917   1.0000
## Neg Pred Value         0.9998   0.9971   0.9983   0.9978   0.9981
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1912   0.1730   0.1619   0.1823
## Detection Prevalence   0.2853   0.1922   0.1769   0.1633   0.1823
## Balanced Accuracy      0.9990   0.9932   0.9937   0.9935   0.9958
```

```r
# Random forest accuracy and out sample error

rf_accuracy <- rf_confm$overall[1]
rf_accuracy
```

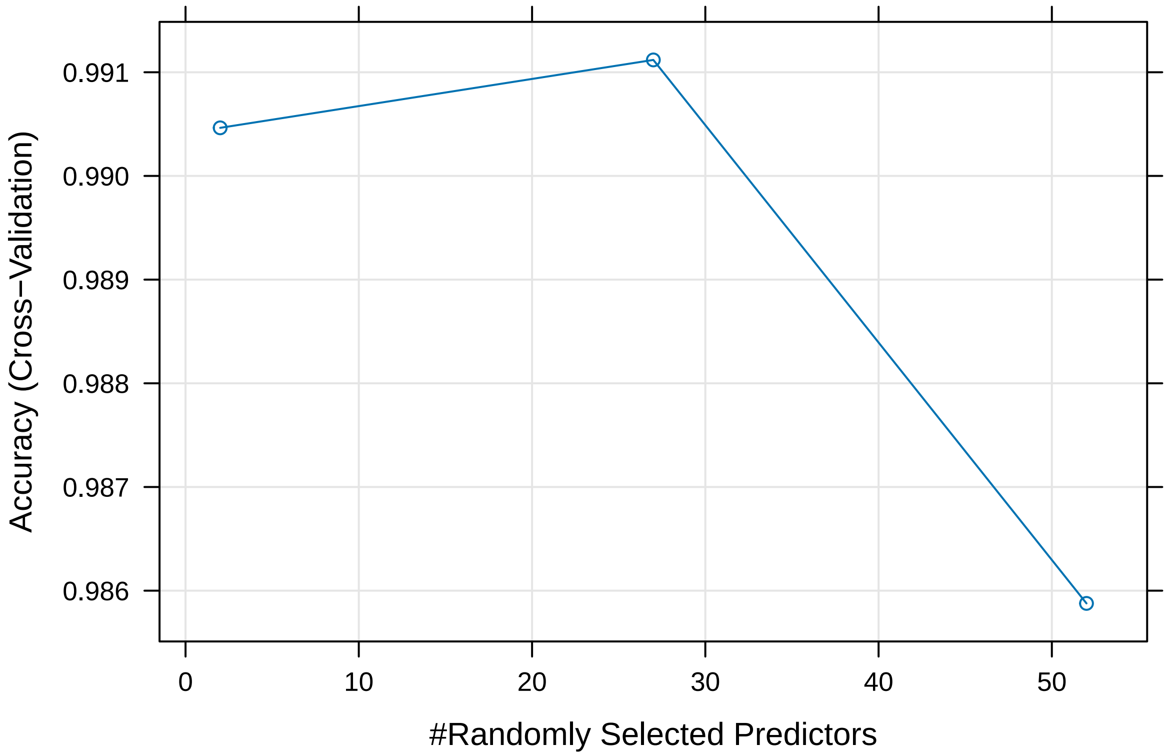
```
##  Accuracy
## 0.9926933
```

```r
rf_outsample <- 1 - rf_accuracy
rf_outsample
```

```
##    Accuracy
## 0.007306712
```

The accuracy rate obtained from the Random Forest is 0.9928632 and the out of sample error rate is 0.007136788.

```r
plot(rf_model)
```



Support Vector Machine

```
# Create support vector machine prediction model with 5 fold cross validation using svmLinear method

support_model <- train(classe ~ ., data = train_data, method = "svmLinear",
                  trControl = valid_control)

# Apply the model to validation set
support_pred <- predict(support_model, valid_data)
support_confm <- confusionMatrix(support_pred, valid_data$classe)
support_confm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1558  149  109   63   59
##          B   27  837   83   42  154
##          C   34   64  796  112   85
##          D   43   18   23  703   52
##          E   12   71   15   44  732
##
## Overall Statistics
##
```

```
##                Accuracy : 0.7861
##                  95% CI : (0.7754, 0.7965)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7277
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9307   0.7349   0.7758   0.7293   0.6765
## Specificity            0.9098   0.9355   0.9393   0.9724   0.9704
## Pos Pred Value         0.8039   0.7323   0.7296   0.8379   0.8375
## Neg Pred Value         0.9706   0.9363   0.9520   0.9483   0.9302
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2647   0.1422   0.1353   0.1195   0.1244
## Detection Prevalence   0.3293   0.1942   0.1854   0.1426   0.1485
## Balanced Accuracy      0.9202   0.8352   0.8576   0.8508   0.8235
```

```r
# Accuracy and out of sample error
support_accuracy <- support_confm$overall[1]
support_accuracy
```

```
##  Accuracy
## 0.7860663
```

```r
support_outsample <- 1 - support_accuracy
support_outsample
```

```
##  Accuracy
## 0.2139337
```

The accuracy rate obtained from the Support Vector Machine model is 0.7860663 and the out of sample error rate is 0.2139337.

Generalized Boosting

```r
# Separate control to use repeated 5 fold cross validation
gnb <- trainControl(method = "repeatedcv", number = 5, verboseIter = FALSE)

# Generalized boosting prediction model with 5 fold repeated
gnb_model <- train(classe ~ ., data = train_data, method = "gbm",
                   trControl = gnb,
                   verbose = FALSE)

# Apply model to the validation set
gnb_pred <- predict(gnb_model, valid_data)
gnb_confm <- confusionMatrix(gnb_pred, valid_data$classe)
gnb_confm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1649   36    0    0    4
##          B   16 1065   33    2    8
##          C    5   38  978   31   21
##          D    2    0   14  919   16
##          E    2    0    1   12 1033
##
## Overall Statistics
##
##                Accuracy : 0.959
##                  95% CI : (0.9537, 0.964)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9482
##
##  Mcnemar's Test P-Value : 1.833e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9851   0.9350   0.9532   0.9533   0.9547
## Specificity            0.9905   0.9876   0.9804   0.9935   0.9969
## Pos Pred Value         0.9763   0.9475   0.9115   0.9664   0.9857
## Neg Pred Value         0.9940   0.9845   0.9900   0.9909   0.9899
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2802   0.1810   0.1662   0.1562   0.1755
## Detection Prevalence   0.2870   0.1910   0.1823   0.1616   0.1781
## Balanced Accuracy      0.9878   0.9613   0.9668   0.9734   0.9758
```

```
# Accuracy and out of sample error
gnb_accuracy <- gnb_confm$overall[1]
gnb_accuracy
```

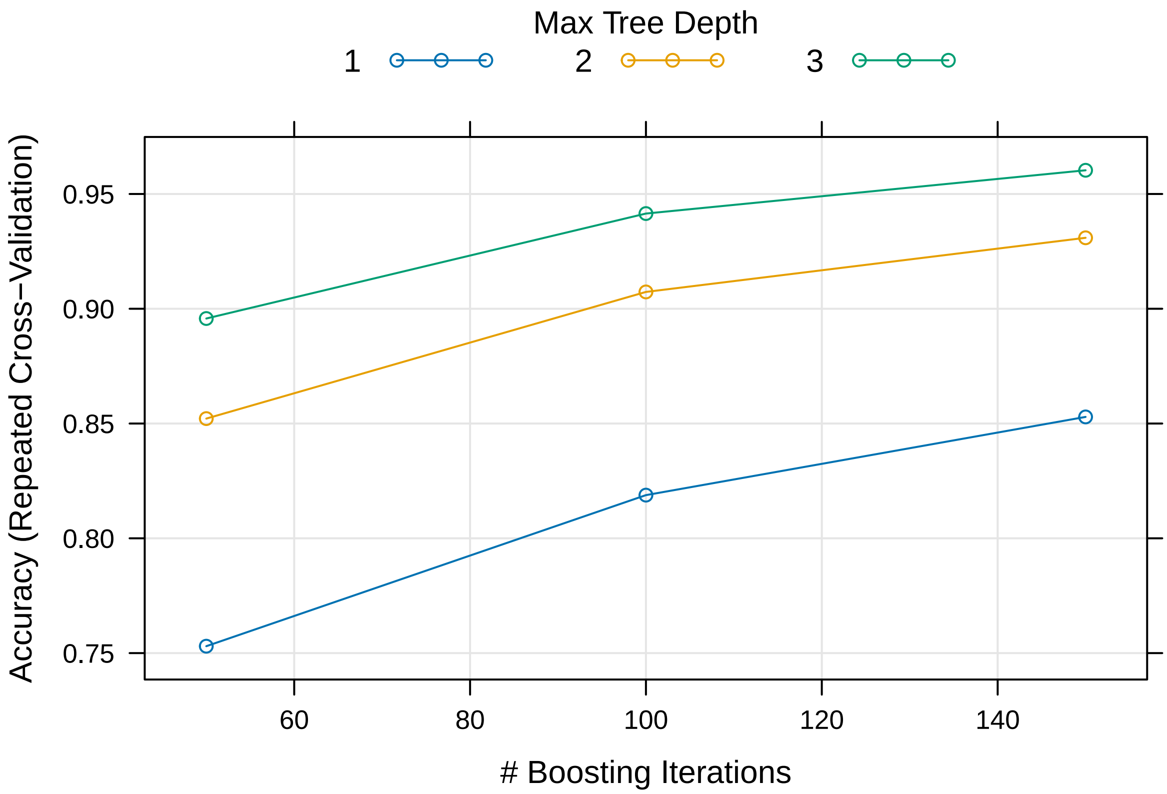
```
##  Accuracy
## 0.9590484
```

```
gnb_outsample <- 1 - gnb_accuracy
gnb_outsample
```

```
##   Accuracy
## 0.04095157
```

The accuracy rate obtained from Generalized Boost model is 0.9632965 and the out of sample error rate is 0.03670348.

```
plot(gnb_model)
```



Summary of prediction model based on their accuracy and out of sample error rate

Summary table of the 4 methods

```
model = c("Decision Tree", "Random Forests", "Support Vector", "Generalized Boosting")
Accuracy <- round(c(tree_accuracy, rf_accuracy, support_accuracy, gnb_accuracy), 4)
Out_of_Sample_Error <- 1 - Accuracy
data.frame(Accuarcy = Accuracy, Out_of_Sample_Error = Out_of_Sample_Error, row.names = model)
```

```
##                        Accuarcy Out_of_Sample_Error
## Decision Tree            0.4997              0.5003
## Random Forests           0.9927              0.0073
## Support Vector           0.7861              0.2139
## Generalized Boosting     0.9590              0.0410
```

The results indicate that Random Forests achieved the highest accuracy (99%) and lowest out-of-sample error (1%), followed by Generalized Boosting (96.33% accuracy, 3.67% error), while Decision Tree and Support Vector Machine exhibited significantly lower performance.

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

Apply Random Forest to the Dataset

```
predict_result <- predict(rf_model, test_data)
predict_result
```

```
##  [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
```