Practical Machine Learning Project

Terhemen Hulugh

2024-03-25

I Overview

In this project, we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

The data consists of a Training data and Testing data.

The goal of this project is to predict the manner in which they did the exercise, that is the "classe" variable in the training set. The dataset was cleaned and the remaining variables were used for the prediction exercise using 3 prediction models. The model with the best accuracy rate was applied to the 20 test cases available in the testing data.

Note: The dataset used in this project is a courtesy of "Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements"

II Load Relevant Libraries

```
rm(list=ls())   # free up memory for the download of the data sets
library(knitr)
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
library(randomForest)
library(corrplot)
```

III Getting, Cleaning and Exploring Data

```
# set the URL for the download of Training and Testing Dataset
urlTrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urlTest  <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# download the Training and Testing datasets
training <- read.csv(url(urlTrain))
testing  <- read.csv(url(urlTest))
dim(training)
```

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

```
dim(testing)
```

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

```
# create a validation dataset from the training dataset
in_train  <- createDataPartition(training$classe, p=0.7, list=FALSE)
train_data <- training[in_train, ]
valid_data  <- training[-in_train, ]
dim(train_data)
```

```
## [1] 13737   160
```

```
dim(valid_data)
```

```
## [1] 5885  160
```

```
#Remove variables with little impact on outcome of Classe
train_data <- train_data[, -c(1:7)]
valid_data <- valid_data[, -c(1:7)]
dim(train_data)
```

```
## [1] 13737   153
```

```
dim(valid_data)
```

```
## [1] 5885  153
```

```
# remove variables with Nearly Zero Variance
NZV <- nearZeroVar(train_data)
train_data <- train_data[, -NZV]
valid_data  <- valid_data[, -NZV]
dim(train_data)
```

```
## [1] 13737    99
```

```
dim(valid_data)
```

```
## [1] 5885   99
```

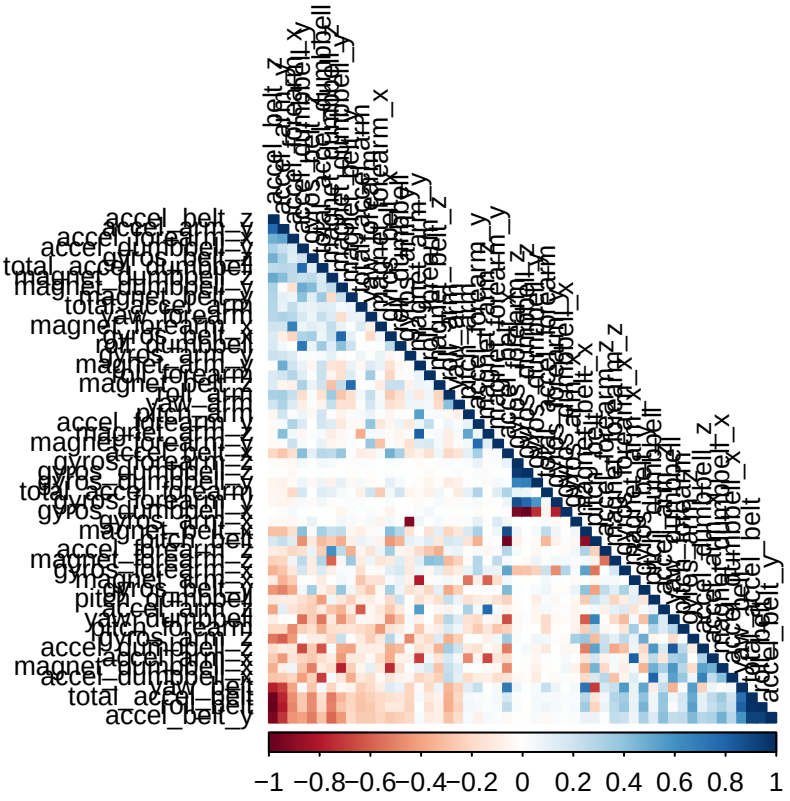
```
#Remove variables containing missing values
train_data<- train_data[, colSums(is.na(train_data)) == 0]
valid_data <- valid_data[, colSums(is.na(valid_data)) == 0]
dim(train_data)
```

```
## [1] 13737    53
```

```
dim(valid_data)
```

```
## [1] 5885   53
```

```
# Plot correlation between variables to explore relationships
cor_matrix <- cor(train_data[, -53])
corrplot(cor_matrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```




```
# Identify highly correlated variables at a cutoff of 70%
highly_correlated = findCorrelation(cor_matrix, cutoff=0.7)
names(train_data)[highly_correlated]
```

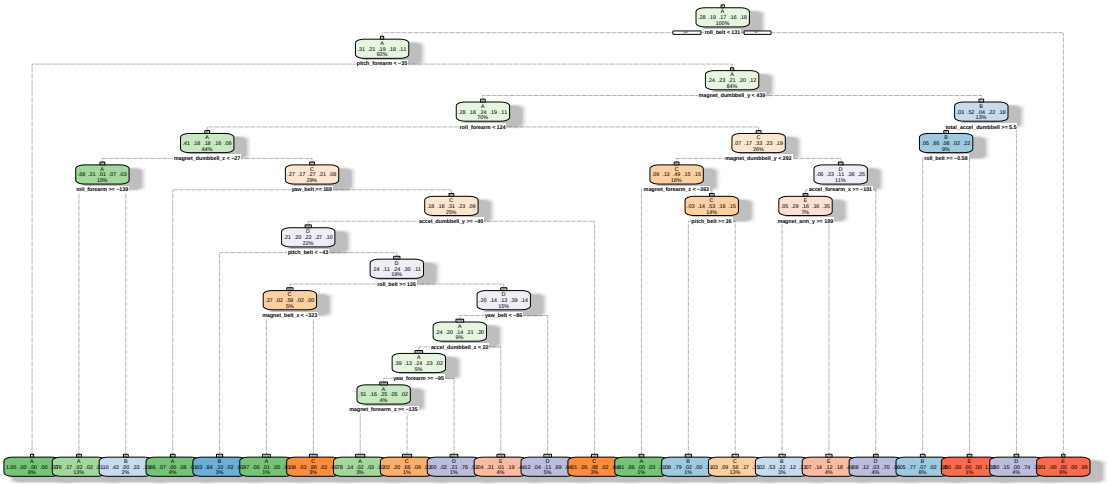
```
##  [1] "accel_belt_z"      "roll_belt"         "accel_belt_y"
##  [4] "total_accel_belt"  "yaw_belt"          "accel_dumbbell_z"
##  [7] "accel_belt_x"      "pitch_belt"        "magnet_dumbbell_x"
## [10] "accel_dumbbell_y"  "magnet_dumbbell_y" "accel_dumbbell_x"
## [13] "accel_arm_x"       "accel_arm_z"       "magnet_arm_y"
## [16] "magnet_belt_z"     "accel_forearm_y"   "gyros_forearm_y"
## [19] "gyros_dumbbell_x"  "gyros_dumbbell_z"  "gyros_arm_x"
```

IV Prediction Model Building

Three methods will be applied in the model building process using the training dataset. The model with the highest accuracy rate will be selected and applied to the testing dataset for the predictions. The methods used for model building are: Decision Tree, Random Forest and Generalized Boosted Model as presented below.

a. Decision Tree Method

```
# model fit
set.seed(12345)
modFitDecTree <- rpart(classe ~ ., data=train_data, method="class")
fancyRpartPlot(modFitDecTree)
```



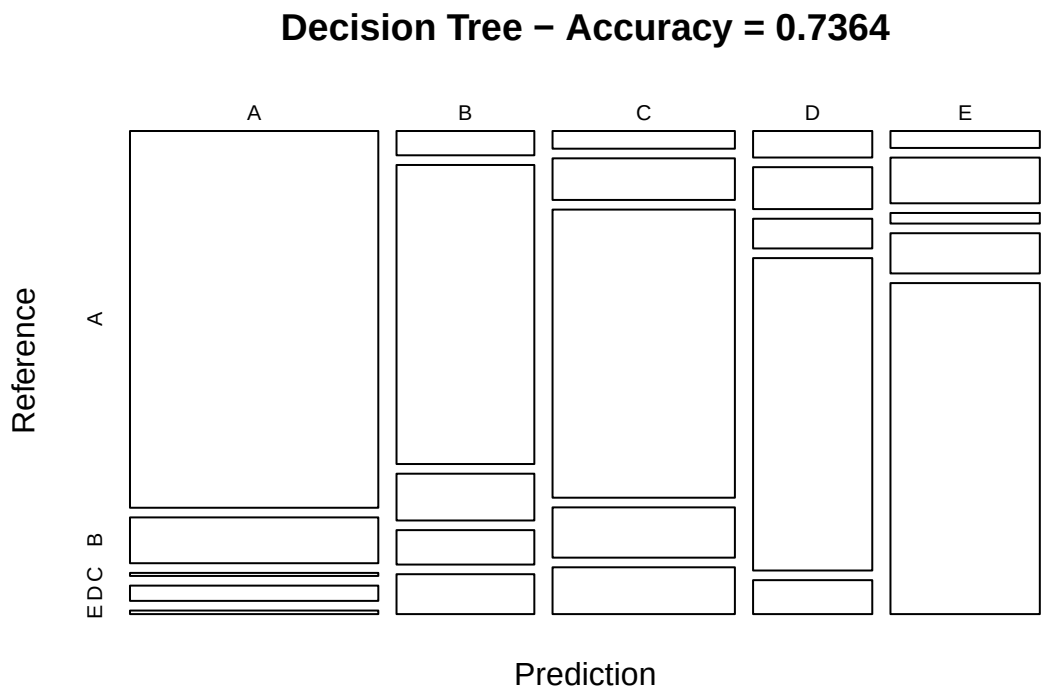
Rattle 2024−Mar−26 10:22:23 TERHEMEN HULUGH

```
# prediction on Validation dataset
predictDecTree <- predict(modFitDecTree, newdata=valid_data, type="class")
confMatDecTree <- confusionMatrix(predictDecTree, as.factor(valid_data$classe))
confMatDecTree
```

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

```
##          A 1480  180   12   60   14
##          B   53  652  102   75   87
##          C   51  120  831  145  135
##          D   50   79   56  589   64
##          E   40  108   25   95  782
##
## Overall Statistics
##
##                Accuracy : 0.7364
##                  95% CI : (0.725, 0.7477)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6662
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8841   0.5724   0.8099   0.6110   0.7227
## Specificity            0.9368   0.9332   0.9072   0.9494   0.9442
## Pos Pred Value         0.8477   0.6729   0.6482   0.7029   0.7448
## Neg Pred Value         0.9531   0.9009   0.9576   0.9257   0.9380
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2515   0.1108   0.1412   0.1001   0.1329
## Detection Prevalence   0.2967   0.1647   0.2178   0.1424   0.1784
## Balanced Accuracy      0.9105   0.7528   0.8586   0.7802   0.8335
```

```r
# plot matrix results
plot(confMatDecTree$table, col = confMatDecTree$byClass,
     main = paste("Decision Tree - Accuracy =",
                  round(confMatDecTree$overall['Accuracy'], 4)))
```



b. Random Forest Method

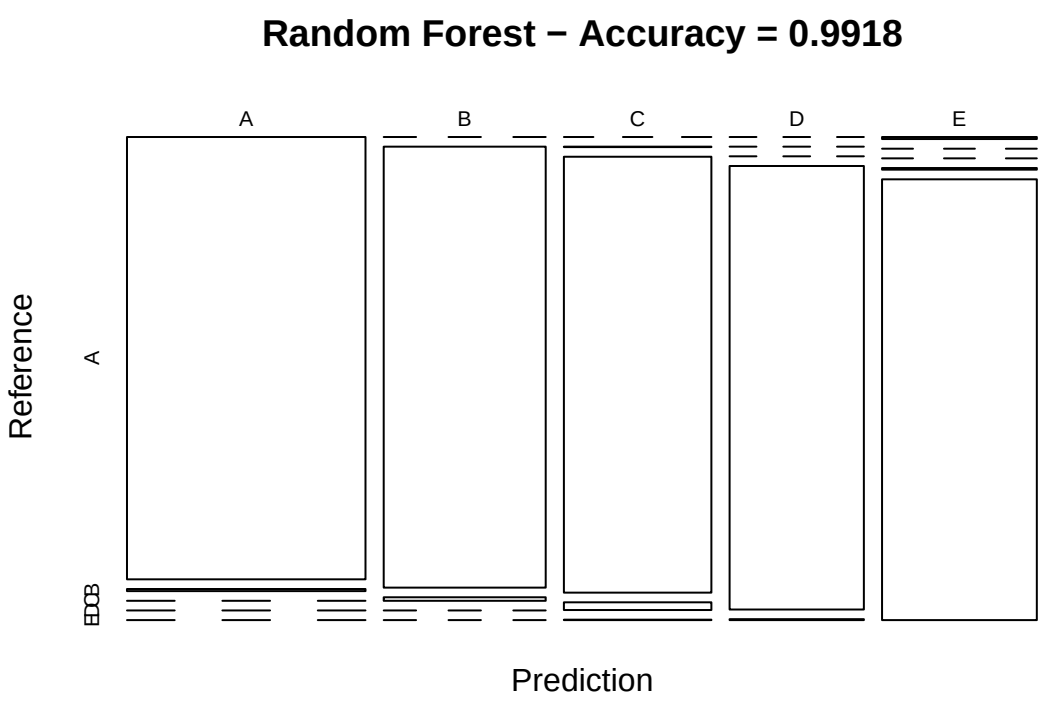
```
# model fit
set.seed(12345)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=train_data, method="rf",
                          trControl=controlRF)
modFitRandForest$finalModel
```

```
##
## 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: 0.81%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3901    4    1    0    0 0.001280082
## B   23 2631    4    0    0 0.010158014
## C    0   19 2371    6    0 0.010434057
## D    0    0   44 2207    1 0.019982238
## E    0    0    2    7 2516 0.003564356
```

```r
# prediction on validation dataset
predictRandForest <- predict(modFitRandForest, newdata=valid_data)
confMatRandForest <- confusionMatrix(predictRandForest, as.factor(valid_data$classe))
confMatRandForest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669    8    0    0    0
##          B    0 1130    9    0    0
##          C    0    1 1017   18    1
##          D    0    0    0  942    2
##          E    5    0    0    4 1079
##
## Overall Statistics
##
##                Accuracy : 0.9918
##                  95% CI : (0.9892, 0.994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9897
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9921   0.9912   0.9772   0.9972
## Specificity            0.9981   0.9981   0.9959   0.9996   0.9981
## Pos Pred Value         0.9952   0.9921   0.9807   0.9979   0.9917
## Neg Pred Value         0.9988   0.9981   0.9981   0.9955   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1920   0.1728   0.1601   0.1833
## Detection Prevalence   0.2850   0.1935   0.1762   0.1604   0.1849
## Balanced Accuracy      0.9976   0.9951   0.9936   0.9884   0.9977
```

```r
# plot matrix results
plot(confMatRandForest$table, col = confMatRandForest$byClass,
     main = paste("Random Forest - Accuracy =",
                  round(confMatRandForest$overall['Accuracy'], 4)))
```



c. Generalized Boosted Model

```
# model fit
set.seed(12345)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM  <- train(classe ~ ., data=train_data, method = "gbm",
                    trControl = controlGBM, verbose = FALSE)
modFitGBM$finalModel
```

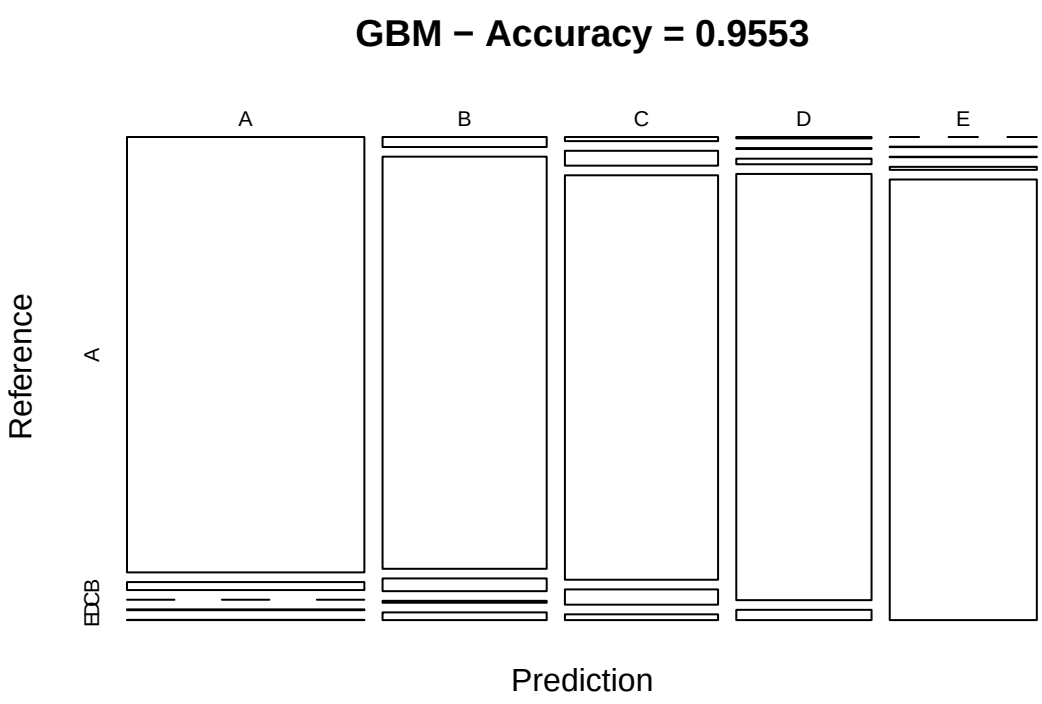
```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

```
# prediction on validation dataset
predictGBM <- predict(modFitGBM, newdata=valid_data)
confMatGBM <- confusionMatrix(predictGBM, as.factor(valid_data$classe))
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1635   30    0    3    1
```

```
##          B   26 1070   33    5   20
##          C   10   36  980   37   14
##          D    3    2   12  912   22
##          E    0    1    1    7 1025
##
## Overall Statistics
##
##                Accuracy : 0.9553
##                  95% CI : (0.9497, 0.9604)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9435
##
##  Mcnemar's Test P-Value : 1.746e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9767   0.9394   0.9552   0.9461   0.9473
## Specificity            0.9919   0.9823   0.9800   0.9921   0.9981
## Pos Pred Value         0.9796   0.9272   0.9099   0.9590   0.9913
## Neg Pred Value         0.9907   0.9854   0.9904   0.9895   0.9882
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2778   0.1818   0.1665   0.1550   0.1742
## Detection Prevalence   0.2836   0.1961   0.1830   0.1616   0.1757
## Balanced Accuracy      0.9843   0.9609   0.9676   0.9691   0.9727
```

```r
# plot matrix results
plot(confMatGBM$table, col = confMatGBM$byClass,
     main = paste("GBM - Accuracy =", round(confMatGBM$overall['Accuracy'], 4)))
```



V. Applying the Selected Model to the Testing Dataset

The results from the above prediction methods show that Random Forest model has the highest accuracy rate with over 99%. Hence, the Random Forest Model will be applied to predict the 20 quiz results using the testing dataset as shown below.

```
predictTEST <- predict(modFitRandForest, newdata=testing)
predictTEST
```

```
##  [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
```