Prediction Assignment

Raheel Shahab

2025-07-04

Note to the Reviewer

This is a second attempt. It appears the first reviewer didn't properly read the assignment and just marked every second option. For example, both the .html and .Rmd files were uploaded to the github repository, but they still marked that one of the files was not available. Same for cross validation. This is unfortunate. I request the reviewer to do a fair marking. I'd be grateful.

Description

This analysis fits a Random Forest classification model to predict exercise quality ("classe" variable) from sensor data. The model is built using the caret and randomForest packages in R.

The Model

Random Forest is used for fitting the data because: * It can handle lots of variables and sort out complicated relationships. As the dataset contains many columns with different types of information, random forest can efficiently handle it. * It uses many trees and combines the their results giving stable and reliable results. * In addition, random forest generally has more accuracy than other algorithms for classification tasks.

Load packages

```
knitr::opts_chunk$set(echo = TRUE)
library(caret, quietly = T, warn.conflicts = F)
library(randomForest, quietly = T, warn.conflicts = F)
```

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

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

```
library(dplyr, quietly = T, warn.conflicts = F)
```

Step 1. Reading and cleaning the data files

```r
#reading and cleaning
data <- read.csv("pml-training.csv", na.strings = c("", "NA", "#DIV/0!"))
data$classe <- as.factor(data$classe)  # Convert classe to factor type
test_data <- read.csv("pml-testing.csv", na.strings = c("", "NA", "#DIV/0!"))

#Columns (variables) with more than 90% observations missing are dropped to reduce the noise
# Remove columns with mostly NAs ( > 90% missing)
na_ratio <- colMeans(is.na(data))
data_clean <- data[, na_ratio < 0.90]

names(data_clean)
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
##  [7] "num_window"           "roll_belt"            "pitch_belt"
## [10] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
## [13] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [16] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [19] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [22] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [25] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [28] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [31] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [34] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [37] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [40] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [43] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [46] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [49] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [52] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [55] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [58] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

```r
# Removing columns that are not useful for prediction
data_clean <- data_clean %>%
  select(-c(X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, num_w
```

Step 2. Splitting into training and validation set, also standardizing/preprocessing

The cleaned dataset is partitioned into training (70%) and validation (30%) sets. Numeric predictor variables are centered (mean subtracted) and scaled (divided by standard deviation) to normailize their ranges.

```r
# Split into training and validation sets
set.seed(123) #for reproducible results

trainIndex <- createDataPartition(data_clean$classe, p = 0.7, list = FALSE)
training <- data_clean[trainIndex, ]
validation <- data_clean[-trainIndex, ]

# Preprocess: center and scale numeric variables
preProc <- preProcess(training[, -ncol(training)], method = c("center", "scale"))
```

```r
training_preprocessed <- predict(preProc, training)
validation_preprocessed <- predict(preProc, validation)
```

Model configuration:

The caret package in R is used with the following configuration: * Two-fold cross-validation is applied to assess model stability and prevent overfitting. Two folds are used because five folds or more are taking a lot of time to train the model with. * The model is built using the default (500) decision trees. * The importance of each predictor is assessed to understand which variables contribute most to the classification task.

Step 3. Training with Random Forrest with cross validation

I tried training entire training dataset with five folds cross validation but it was taking a lot of time. So, attempted a sample of 7000 and also the entire dataset with two folds cross validation. The results were almost the same.

```r
# Uncomment the below to train random forest again
# model_rf <- train(classe ~ .,
#                    data = training_preprocessed, #%>% slice_sample(n = 7000), #training takes forever.
#                    method = "rf",
#                    trControl = trainControl(method = "cv", number = 2, verboseIter = T),
#                    #ntree = 100,
#                    importance = TRUE)

#saveRDS(model_rf, file = 'model_rf_all_n.rds')
#saveRDS(model_rf, file = 'model_rf_7000_n.rds')

model_rf <- readRDS('model_rf_all_n.rds') #comment this out.
```

Step 4. Evaluating the model

```r
# Evaluate on validation set
pred_rf <- predict(model_rf, newdata = validation_preprocessed)
conf_mat <- confusionMatrix(pred_rf, validation_preprocessed$classe)

conf_mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    5    0    0    0
##          B    1 1128    5    0    0
##          C    0    6 1018    9    4
##          D    0    0    3  955    4
##          E    0    0    0    0 1074
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9937
##                  95% CI : (0.9913, 0.9956)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.992
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9903   0.9922   0.9907   0.9926
## Specificity            0.9988   0.9987   0.9961   0.9986   1.0000
## Pos Pred Value         0.9970   0.9947   0.9817   0.9927   1.0000
## Neg Pred Value         0.9998   0.9977   0.9983   0.9982   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1917   0.1730   0.1623   0.1825
## Detection Prevalence   0.2851   0.1927   0.1762   0.1635   0.1825
## Balanced Accuracy      0.9991   0.9945   0.9941   0.9946   0.9963
```

Step. 5 Model Accuracy and Out-of-sample Error

```r
cat("Random Forest Model Accuracy:", round(conf_mat$overall['Accuracy'], 4), "\n")
```

```
## Random Forest Model Accuracy: 0.9937
```

```r
cat("Out-of-sample error estimate:", round(1 - conf_mat$overall['Accuracy'], 4), "\n\n")
```

```
## Out-of-sample error estimate: 0.0063
```

```r
# Variable importance
var_imp <- varImp(model_rf)

print(var_imp)
```

```
## rf variable importance
##
##   variables are sorted by maximum importance across the classes
##   only 20 most important variables shown (out of 52)
##
##                        A     B     C     D      E
## roll_belt         75.219 85.33 78.69 80.39 100.00
## pitch_belt        25.170 86.95 56.68 39.82  33.88
## pitch_forearm     53.703 60.61 82.56 48.11  52.36
## magnet_dumbbell_y 61.691 60.00 77.56 57.84  48.53
## magnet_dumbbell_z 75.306 57.70 73.80 49.44  48.98
## yaw_belt          59.394 56.21 63.07 67.36  43.36
## accel_forearm_x   17.013 34.09 30.44 46.47  30.52
```

```
## roll_forearm      40.291 29.99 35.54 24.23  25.76
## yaw_arm           36.603 23.59 25.73 28.37  16.19
## accel_dumbbell_y  31.550 28.12 36.08 25.86  30.85
## gyros_belt_z      19.138 26.59 27.46 18.83  33.70
## gyros_dumbbell_y  31.156 22.19 32.00 19.46  18.27
## magnet_belt_z     18.772 30.99 19.61 27.60  24.20
## gyros_arm_y       22.276 26.84 19.99 28.52  17.95
## accel_dumbbell_z  21.773 26.70 20.73 26.03  28.27
## magnet_arm_z      14.738 28.09 21.86 18.84  16.11
## roll_dumbbell     18.558 26.74 19.49 21.59  27.41
## magnet_belt_x      8.261 26.72 21.81 13.32  16.82
## magnet_belt_y     13.758 25.95 23.45 18.86  21.06
## roll_arm          11.942 25.01 17.02 18.54  11.74
```

Step 6. Prediction on the test set

```r
dim(test_data)
```

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

```r
dim(data_clean) #training
```

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

```r
test_data_clean <- test_data[, names(data_clean)[-ncol(data_clean)]]  # Match training columns
test_data_preprocessed <- predict(preProc, test_data_clean)

final_predictions <- predict(model_rf, newdata = test_data_preprocessed)
cat("\nPredictions for test cases:\n", as.character(final_predictions))
```

```
##
## Predictions for test cases:
##  B A B A A E D B A A B C B A E E A B B B
```