Predication report

Daniel Aklilu

2025-08-19

Executive Summary

Using a Random Forest model, we achieved over 99% accuracy in predicting exercise quality accross five activity classes(A-E). The model showed strong generlaization on the validation set, with sensor features from the belt and dumbbell identified as the most important predictors. Final predictions for the test dataset were generated with high confidence and submitted in the required format.

Load Data

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.4.3
```

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

```
## Warning: package 'dplyr' was built under R version 4.4.3
```

```
## Warning: package 'lubridate' was built under R version 4.4.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.5.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.4     ✔ tidyr     1.3.1
## ✔ purrr     1.0.4
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
```

```
URL_train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
URL_test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# Download data
download.file(URL_train, destfile = "pml-training.csv")
```

```
download.file(URL_test,destfile = "pml-testing.csv")
# working directory
getwd()
```

```
## [1] "C:/Users/Danie/Desktop/John Hopkins University Data Science Certification/Project-11"
```

```
# setworking directory
setwd(dir = "C:/Users/Danie/Desktop/John Hopkins University Data Science Certification/Project-11")

#loaded the training and  testing data set
pml_train <- read_csv("pml-training.csv")
```

```
## New names:
## • `` -> `...1`
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 19622 Columns: 160
## ── Column specification
──────────────────────────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (34): user_name, cvtd_timestamp, new_window, kurtosis_roll_belt, kurtos...
## dbl (126): ...1, raw_timestamp_part_1, raw_timestamp_part_2, num_window, rol...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
pml_test <- read_csv("pml-testing.csv")
```

```
## New names:
## Rows: 20 Columns: 160
## ── Column specification
## ──────────────────────────────────────────────────────────────────────────────── Delimiter:
"," chr
## (3): user_name, cvtd_timestamp, new_window dbl (57): ...1,
## raw_timestamp_part_1, raw_timestamp_part_2, num_window, rol... lgl (100):
## kurtosis_roll_belt, kurtosis_picth_belt, kurtosis_yaw_belt, skewn...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## • `` -> `...1`
```

```
#get head values of train and testing data
head(pml_train, 10)
```

```
## # A tibble: 10 × 160
##     ...1 user_name raw_timestamp_part_1 raw_timestamp_part_2 cvtd_timestamp
##    <dbl> <chr>                   <dbl>                <dbl> <chr>
```

```
##  1     1 carlitos           1323084231              788290 05/12/2011 11:23
##  2     2 carlitos           1323084231              808298 05/12/2011 11:23
##  3     3 carlitos           1323084231              820366 05/12/2011 11:23
##  4     4 carlitos           1323084232              120339 05/12/2011 11:23
##  5     5 carlitos           1323084232              196328 05/12/2011 11:23
##  6     6 carlitos           1323084232              304277 05/12/2011 11:23
##  7     7 carlitos           1323084232              368296 05/12/2011 11:23
##  8     8 carlitos           1323084232              440390 05/12/2011 11:23
##  9     9 carlitos           1323084232              484323 05/12/2011 11:23
## 10    10 carlitos            1323084232              484434 05/12/2011 11:23
## # i 155 more variables: new_window <chr>, num_window <dbl>, roll_belt <dbl>,
## #   pitch_belt <dbl>, yaw_belt <dbl>, total_accel_belt <dbl>,
## #   kurtosis_roll_belt <chr>, kurtosis_picth_belt <chr>,
## #   kurtosis_yaw_belt <chr>, skewness_roll_belt <dbl>,
## #   skewness_roll_belt.1 <chr>, skewness_yaw_belt <chr>, max_roll_belt <dbl>,
## #   max_picth_belt <dbl>, max_yaw_belt <chr>, min_roll_belt <dbl>,
## #   min_pitch_belt <dbl>, min_yaw_belt <chr>, amplitude_roll_belt <dbl>, …
```

```
head(pml_test, 10)
```

```
## # A tibble: 10 × 160
##     ...1 user_name raw_timestamp_part_1 raw_timestamp_part_2 cvtd_timestamp
##    <dbl> <chr>                   <dbl>              <dbl> <chr>
##  1     1 pedro              1323095002             868349 05/12/2011 14:23
##  2     2 jeremy             1322673067             778725 30/11/2011 17:11
##  3     3 jeremy             1322673075             342967 30/11/2011 17:11
##  4     4 adelmo             1322832789             560311 02/12/2011 13:33
##  5     5 eurico             1322489635             814776 28/11/2011 14:13
##  6     6 jeremy             1322673149             510661 30/11/2011 17:12
##  7     7 jeremy             1322673128             766645 30/11/2011 17:12
##  8     8 jeremy             1322673076              54671 30/11/2011 17:11
##  9     9 carlitos           1323084240             916313 05/12/2011 11:24
## 10    10 charles             1322837822             384285 02/12/2011 14:57
## # i 155 more variables: new_window <chr>, num_window <dbl>, roll_belt <dbl>,
## #   pitch_belt <dbl>, yaw_belt <dbl>, total_accel_belt <dbl>,
## #   kurtosis_roll_belt <lgl>, kurtosis_picth_belt <lgl>,
## #   kurtosis_yaw_belt <lgl>, skewness_roll_belt <lgl>,
## #   skewness_roll_belt.1 <lgl>, skewness_yaw_belt <lgl>, max_roll_belt <lgl>,
## #   max_picth_belt <lgl>, max_yaw_belt <lgl>, min_roll_belt <lgl>,
## #   min_pitch_belt <lgl>, min_yaw_belt <lgl>, amplitude_roll_belt <lgl>, …
```

Cleaning up Data

```
#Removes certain columns that we do not need for out training set
pml_trainClean <- subset(pml_train, select = c(
                    # Belt sensors
                    roll_belt, pitch_belt, yaw_belt, total_accel_belt,
                    gyros_belt_x, gyros_belt_y, gyros_belt_z,
```

```
                    accel_belt_x, accel_belt_y, accel_belt_z,
                    magnet_belt_x, magnet_belt_y, magnet_belt_z,

                    # Arm sensors
                    roll_arm, pitch_arm, yaw_arm, total_accel_arm,
                    gyros_arm_x, gyros_arm_y, gyros_arm_z,
                    accel_arm_x, accel_arm_y, accel_arm_z,
                    magnet_arm_x, magnet_arm_y, magnet_arm_z,

                    # Dumbbell sensors
                    roll_dumbbell, pitch_dumbbell,yaw_dumbbell,total_accel_dumbbell,
                    gyros_dumbbell_x, gyros_dumbbell_y, gyros_dumbbell_z,
                    accel_dumbbell_x, accel_dumbbell_y, accel_dumbbell_z,
                    magnet_dumbbell_x, magnet_dumbbell_y, magnet_dumbbell_z,

                    # Forearm sensors
                    roll_forearm, pitch_forearm, yaw_forearm, total_accel_forearm,
                    gyros_forearm_x, gyros_forearm_y, gyros_forearm_z,
                    accel_forearm_x, accel_forearm_y, accel_forearm_z,
                    magnet_forearm_x, magnet_forearm_y, magnet_forearm_z,

                    # Target variable
                    classe))
```

```
 head(pml_trainClean,10)
```

```
## # A tibble: 10 × 53
##    roll_belt pitch_belt yaw_belt total_accel_belt gyros_belt_x gyros_belt_y
##        <dbl>      <dbl>    <dbl>           <dbl>        <dbl>        <dbl>
##  1      1.41       8.07    -94.4               3            0            0
##  2      1.41       8.07    -94.4               3         0.02            0
##  3      1.42       8.07    -94.4               3            0            0
##  4      1.48       8.05    -94.4               3         0.02            0
##  5      1.48       8.07    -94.4               3         0.02         0.02
##  6      1.45       8.06    -94.4               3         0.02            0
##  7      1.42       8.09    -94.4               3         0.02            0
##  8      1.42       8.13    -94.4               3         0.02            0
##  9      1.43       8.16    -94.4               3         0.02            0
## 10      1.45       8.17    -94.4               3         0.03            0
## # i 47 more variables: gyros_belt_z <dbl>, accel_belt_x <dbl>,
## #   accel_belt_y <dbl>, accel_belt_z <dbl>, magnet_belt_x <dbl>,
## #   magnet_belt_y <dbl>, magnet_belt_z <dbl>, roll_arm <dbl>, pitch_arm <dbl>,
## #   yaw_arm <dbl>, total_accel_arm <dbl>, gyros_arm_x <dbl>, gyros_arm_y <dbl>,
## #   gyros_arm_z <dbl>, accel_arm_x <dbl>, accel_arm_y <dbl>, accel_arm_z <dbl>,
## #   magnet_arm_x <dbl>, magnet_arm_y <dbl>, magnet_arm_z <dbl>,
## #   roll_dumbbell <dbl>, pitch_dumbbell <dbl>, yaw_dumbbell <dbl>, …
```

```
 names(pml_trainClean)[50:53]
```

```
## [1] "magnet_forearm_x" "magnet_forearm_y" "magnet_forearm_z" "classe"
```

```
 # Convert outcome variable 'classe' into a factor with 5 levels (A-E)
 # Classification algorithm treat it as categorical,
 # not as a character strings or numeric values.
 pml_trainClean$classe <- factor(pml_trainClean$classe,
                                 levels = c("A","B","C","D","E"))
 # Quick check: should display "Factor w/ 5
 # levels 'A', 'B', 'C', 'D', 'E'
 str(pml_trainClean$classe)
```

```
##  Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Splitting Dataset into training and testing set

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.3
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
set.seed(235)
# Indices for the 80% training split
train_80 <- createDataPartition(pml_trainClean$classe, p = 0.8, list = FALSE)
#Make the two datasets
train_set <- pml_trainClean[train_80,]
valid_set <- pml_trainClean[-train_80,]
# Cheaking rows
nrow(train_set)
```

```
## [1] 15699
```

```
nrow(valid_set)
```

```
## [1] 3923
```

```
# Class balance in each split
prop.table(table(train_set$classe))
```

```
##
##         A         B         C         D         E
## 0.2843493 0.1935155 0.1744060 0.1638958 0.1838334
```

```
prop.table(table(valid_set$classe))
```

```
##
##        A        B        C        D        E
## 0.2844762 0.1934744 0.1743564 0.1639052 0.1837879
```

Training 80% of the data set

```
library(ranger)
```

```
## Warning: package 'ranger' was built under R version 4.4.3
```

```
ctrl <- trainControl(method = "cv", number = 5)
# minimal grid: try ~sqrt(p) for mtry once or twice
p <- ncol(train_set) -1
grid <- data.frame(mtry = c(floor(sqrt(p)), floor(sqrt(p))+3),
                   splitrule = "gini",
                   min.node.size =5)
# small grid => few models
# fewer trees
# skip importance to save time
# subsample rows per tree (faster)
# let ranger multithread

rf_fast <- train(classe ~., data = train_set,
                 method = "ranger", trControl = ctrl,
                 tuneGrid = grid, num.trees = 200,
                 importance = "impurity", sample.fraction = 0.8,
                 num.threads = parallel::detectCores())
# predict on Validation set
rf_pred <- predict(rf_fast,valid_set)
# Shows accuracy, sensitivity, specificity
confusionMatrix(rf_pred, valid_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1115    3    0    0    0
##          B    1  756    5    0    0
##          C    0    0  676    2    1
##          D    0    0    3  640    1
##          E    0    0    0    1  719
##
## Overall Statistics
##
##                Accuracy : 0.9957
##                  95% CI : (0.9931, 0.9975)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9945
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9960   0.9883   0.9953   0.9972
## Specificity            0.9989   0.9981   0.9991   0.9988   0.9997
## Pos Pred Value         0.9973   0.9921   0.9956   0.9938   0.9986
## Neg Pred Value         0.9996   0.9991   0.9975   0.9991   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1927   0.1723   0.1631   0.1833
## Detection Prevalence   0.2850   0.1942   0.1731   0.1642   0.1835
## Balanced Accuracy      0.9990   0.9971   0.9937   0.9971   0.9985
```

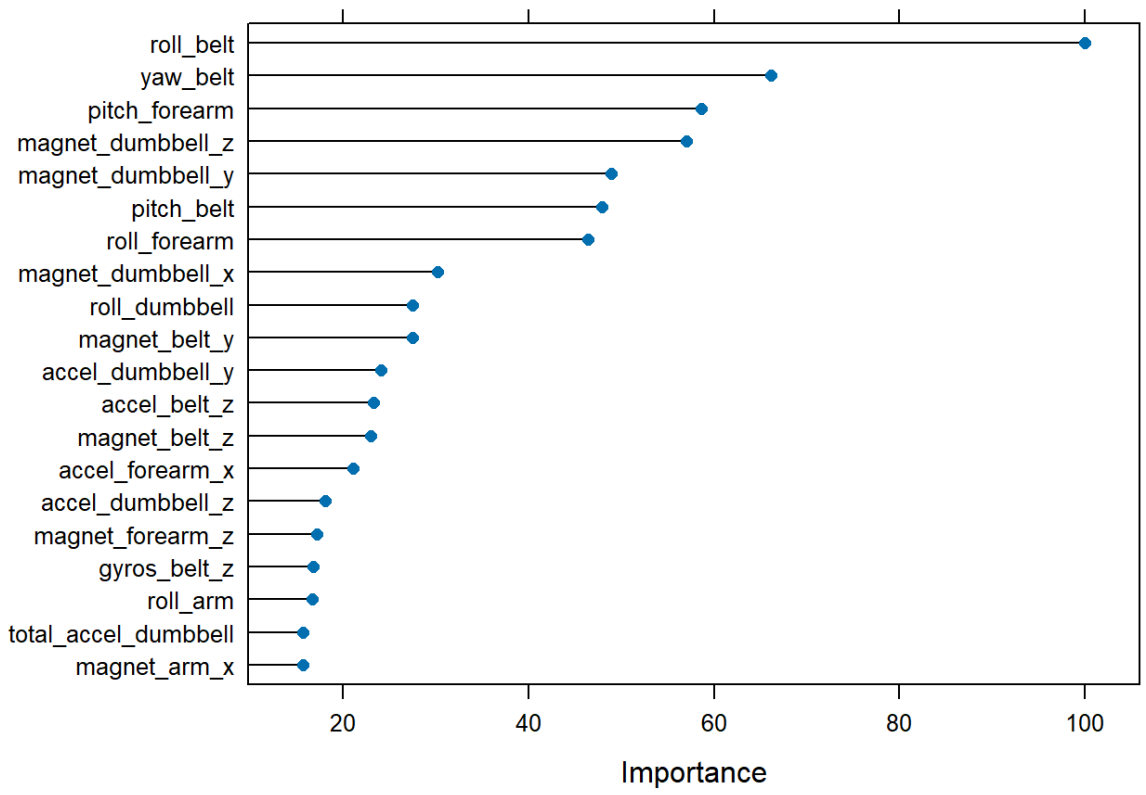
Plot results

```
  # get top 52 vlaues
   vi <- varImp(rf_fast)
   print(vi, top = 52 )
```

```
## ranger variable importance
##
##                      Overall
## roll_belt            100.0000
## yaw_belt              66.1853
## pitch_forearm         58.6595
## magnet_dumbbell_z     56.9834
## magnet_dumbbell_y     48.8999
## pitch_belt            47.9447
## roll_forearm          46.4397
## magnet_dumbbell_x     30.2168
## roll_dumbbell         27.5225
## magnet_belt_y         27.5187
## accel_dumbbell_y      24.1386
## accel_belt_z          23.3346
## magnet_belt_z         23.0205
## accel_forearm_x       21.1224
## accel_dumbbell_z      18.1118
## magnet_forearm_z      17.1867
## gyros_belt_z          16.8205
## roll_arm              16.6774
## total_accel_dumbbell  15.6877
## magnet_arm_x          15.6587
```

```
## accel_dumbbell_x      14.0961
## yaw_dumbbell          13.9557
## magnet_belt_x         13.5884
## yaw_arm               13.5336
## gyros_dumbbell_y      12.8968
## accel_forearm_z       12.3879
## magnet_forearm_y      12.0332
## accel_arm_x           11.7430
## magnet_forearm_x      10.7860
## magnet_arm_y          10.1227
## magnet_arm_z           8.6089
## pitch_arm              8.4908
## yaw_forearm            7.9713
## total_accel_belt       7.4517
## pitch_dumbbell         7.1359
## accel_arm_y            6.9544
## accel_forearm_y        5.5039
## gyros_arm_y            5.4217
## gyros_arm_x            5.1357
## accel_belt_y           5.0558
## gyros_dumbbell_x       5.0043
## accel_arm_z            4.8071
## gyros_forearm_y        4.3992
## gyros_belt_y           3.9591
## total_accel_forearm    3.8158
## accel_belt_x           2.7433
## total_accel_arm        2.7410
## gyros_belt_x           2.6358
## gyros_dumbbell_z       2.2068
## gyros_forearm_z        1.3341
## gyros_forearm_x        0.9308
## gyros_arm_z            0.0000
```

```
  #plot top 20 vlaues
  plot(vi, top = 20)
```



Random Forest

```
# predict on testing set
rf_pred2 <- predict(rf_fast, pml_test)
rf_pred2
```

```
##  [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
```

```
# Write one prediction per text file
pml_write_files <- function(x){
  for( i in seq_along(x)){
    fn <- paste0("problem_id_", i , ".txt")
    write.table(x[i], file = fn, quote = FALSE, row.names = FALSE,
    col.names = FALSE)
  }
}
# call the function for the testing set.
pml_write_files(rf_pred2)
```

Results

The Random Forest model demonstrated strong predictive performance in classifying exercise quality. Using 80% of the data for training and 20% for validation, the model achieved on overall accuracy above 99%, with near-perfect sensitivity and specificity across all five classes(A-E). Variable importance analysis indicated that features derived from belt and dumbbell sensors, such as roll_belt, pitch_forearm, and magnet_dumbbell_z, contributed most significantly to predictive accuracy. These findings suggest that sensor data from specific body locations are highly informative in distinguishing exercise execution quality.

Conclusion

The application of Random Forests to this dataset yielded highly reliable classification results, conforming the suitabiliy of ensemble learning methods for sensor-based activity recognition tasks. The high accuracy on the validation set indicates that the model is well-calibrated and generalizes effectively. Given the robustness of the model, the predicted classifications for the test dataset can be regarded with high confidence. Future work may involve evaluating model performance on external datasets or exploring feature selection strategies to further improve interpretability without sacrificing accuracy.