MET529 - Term Paper (Classification Algorithms to detect human activity)

Anonymous

2024-11-28

Introduction

Wearable technology has been progressively incorporated into the healthcare sector in recent years, revolutionizing both professional medical practice and personal health monitoring. With applications ranging from illness prevention, rehabilitation, and physical therapy to areas far beyond fitness tracking, devices like Fitbit, Nike FuelBand, and Jawbone Up have made it possible to gather enormous volumes of data on human activities. The quantified self movement, in which people utilize self-monitoring devices to collect comprehensive data about different facets of their lives, is consistent with this trend. Wearable technology has a significant potential impact on the healthcare industry. By enabling the early identification of problems like sedentary behavior, irregular exercise habits, or symptoms associated with chronic illnesses, continuous physical activity monitoring can offer vital insights into an individual's health status (Rodgers et al., 2019; Piwek et al., 2016). Furthermore, wearable technology can now quantify the quality of activity in addition to activity volume thanks to developments in data analytics, particularly the application of machine learning and classification algorithms. This is a useful application for both health professionals and tech enthusiasts (Mukhopadhyay, 2014; Wang et al., 2005).

Processing and understanding data from wearables could be greatly aided by applied analytics using R, one of the top statistical computing environments, particularly in healthcare settings. Due to its broad ecosystem of machine learning libraries and its strong data manipulation and visualization capabilities, R-based analytics is well regarded and is especially well-suited for tasks involving activity recognition (Wickham, 2019; Kuhn & Johnson, 2019). In this project, accelerometer data obtained from wearable sensors during exercise is analyzed using R's applied analytics tools. This information, which was gathered from participants' belts, forearms, arms, and dumbbells while they were executing barbell lifts, offers a chance to investigate classification algorithms that can forecast how these activities will be executed. This project illustrates the use of R for creating predictive models that have immediate uses in healthcare, like remote patient monitoring and customized workout feedback, by distinguishing proper and wrong movements in exercises (Bejarano et al., 2014).

In the medical field, evaluating the quality of an activity is especially helpful for physical therapy and rehabilitation, as proper form is essential to patient recovery and injury prevention. For example, physical therapists may be able to remotely monitor their clients' compliance with recommended rehabilitation regimens by employing wearable sensors to determine whether the patient is executing exercises correctly. This is especially important in light of the COVID-19 epidemic, which has prompted a rise in the use of remote monitoring tools and telehealth services. R's classification algorithms have additional potential uses in preventative health, as seeing abnormal movement patterns could help with the early detection of musculoskeletal conditions. The research aims to show how R-based applied analytics may improve health care applications by making it possible to analyze wearable sensor data accurately and meaningfully, laying the groundwork for more individualized and responsive health interventions.

It is now feasible to get a significant quantity of data on personal activity at a reasonable cost by using gadgets like Fitbit, Nike FuelBand, and Jawbone Up. These gadgets are a part of the quantified self movement, which

is a group of enthusiasts that measure themselves on a regular basis for a variety of reasons, such as being tech geeks, finding patterns in their behavior, or improving their health. People frequently measure how much of a certain task they perform, but they hardly ever measure how well they perform it. Six individuals' belt, forearm, arm, and dumbbell accelerometer data will be used in this experiment. They were instructed to lift barbells five various ways, both correctly and incorrectly.

People's activity habits will be predicted by the study. In the train set, this is the 'classes' variable. The five different 'classe' factors in this dataset are: * Exactly according to the specification (Class A) * Throwing the elbows to the front (Class B) * Lifting the dumbbell only halfway (Class C) * Lowering the dumbbell only halfway (Class D) * Throwing the hips to the front (Class E). The data used in the analysis can be downloaded here.

Analysis

Libraries

The CRAN libraries **readr**, **rsample**, **dplyr**, **nnet**, **rpart**, **caret** and **randomForest** are used in this project.

When importing data into R, especially when working with huge datasets, the **readr** library is crucial. It was created by Hadley Wickham and is a component of the tidyverse ecosystem. It is performance tuned, allowing for quick and effective data reading methods like read_csv and read_tsv. These functions improve readability and interoperability with other tidyverse packages by loading data straight into tibbles, a contemporary substitute for R's base data frames (Wickham, 2019). In projects like this one, where accuracy in data ingestion affects later stages of analysis, readr's functionality—which handles missing values, specifies column types, and provides relevant error messages—minimizes data import problems. In health analytics, where data volumes might be significant, readr's structure also makes it more memory-efficient and able to handle large datasets without stuttering, according to Wickham (2019).

Specifically made for data resampling, the **rsample** package provides a range of functions for performing cross-validation and dividing data into training and testing groups. Because it has tools for Monte Carlo cross-validation, k-fold cross-validation, and bootstrapping, it makes it possible to test machine learning models rigorously (Kuhn & Johnson, 2019). Resampling prevents overfitting and guarantees that models generalize well to fresh data, which is crucial in applications related to health where model accuracy is crucial, making this feature crucial for applied analytics projects. For example, the rsample's initial_split effectively divides the dataset, enabling various analyses without data leakage between the training and testing stages. This package is widely cited in academic and business contexts as a crucial tool for statistical modeling and validation in R, and its design is in line with best practices in model evaluation (Kuhn & Johnson, 2019).

Another tidyverse package, **dplyr**, is essential for data manipulation because it offers a "grammar of data manipulation" that makes difficult data wrangling jobs easier (Wickham , 2019). The main features of dplyr, including filter, select, mutate, summarize, and arrange, let users to employ pipes (%>%) to link actions in an understandable manner. Because of this, dplyr is ideal for preprocessing stages in machine learning processes, where effective data gathering, transformation, and filtering are essential. Dplyr's ability to efficiently handle huge datasets in health data analysis ensures that only pertinent characteristics are kept, which expedites the training of downstream models. According to research, dplyr's data manipulation syntax has gained widespread adoption because it makes R scripts easier to read, improves reproducibility, and simplifies coding (Wickham, 2019).

Multinomial log-linear models and basic neural networks can be implemented using the **nnet** package, a neural network package in R (Venables & Ripley, 2002). Single-hidden-layer neural networks, which are useful for classification tasks like activity form prediction in this project, can be specified by users. Even while nnet is more straightforward than deep learning frameworks, it works well for rapid model testing and training, particularly on datasets of a moderate size. For example, multinomial logistic regression, which may be applied to classification problems with several categories, commonly uses the multinom function

within nnet. Venables and Ripley (2002) state that nnet is still widely used in the R community due to its user-friendliness and computational efficiency, which makes it a dependable choice for beginning neural network applications.

One of the most interpretable machine learning models is the decision tree, which may be constructed using functions in the **rpart** package (Recursive Partitioning and Regression Trees). Because they provide transparent decision criteria that are simple for both patients and practitioners to understand, decision trees are very useful in the healthcare industry (Therneau & Atkinson, 1997). By making it possible to build decision tree models that categorize exercise movements using sensor data, rpart facilitates classification tasks in this research. Users can balance the trade-off between model simplicity and accuracy by adjusting tree complexity using the package's rpart function. The predictive potential of decision trees is increased by their compatibility with ensemble techniques like boosting and random forests. The broad use of rpart in industry and scholarly research highlights its versatility and resilience across a range of fields, including health care analytics (Therneau & Atkinson, 1997).

By combining data preparation, model training, tuning, and evaluation, the **caret** (Classification and Regression Training) package in R simplifies the machine learning process (Kuhn, 2008). It has features for automatic model adjustment using grid search and cross-validation, and it supports more than 200 machine learning methods. From linear models to intricate techniques like support vector machines and ensemble approaches, Caret streamlines the administration of several models while offering uniformity and effectiveness across various machine learning workflows. Caret is especially helpful in health care analytics because it makes it possible to quickly prototype and compare models, which is critical when working with patient data where precision is vital (Kuhn, 2013). Because it may standardize model evaluation and integrate with resampling techniques to reduce overfitting, caret is frequently cited in academic literature as a crucial tool for predictive modeling in R (Kuhn, 2008).

The random forest approach, an ensemble learning technique that creates several decision trees and combines their output to increase classification and regression accuracy, can be implemented well with R's **randomForest** package (Breiman, 2001). This package is highly regarded for its resilience and capacity to manage intricate, high-dimensional information, which makes it ideal for uses like wearable sensor data-based human activity detection. In order to decrease overfitting and improve model stability, random forests build a large number of decision trees during training and output the mean prediction (for regression) or the mode of their predictions (for classification). Random forests provide insights into feature importance, which aids in identifying the most important variables influencing predictions, such as particular accelerometer readings in this project. This is especially useful in health analytics, where model interpretability and accuracy are crucial (Rodgers et al., 2019).

```
suppressPackageStartupMessages(library(readr))
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(nnet))
suppressPackageStartupMessages(library(caret))
suppressPackageStartupMessages(library(rsample))
suppressPackageStartupMessages(library(rpart))
suppressPackageStartupMessages(library(randomForest))
```

Data

The data will be downloaded and read into R if it is not accessible. Data will be separated into **test** and **train** categories. Below is a rough snapshot of the data. Of its 160 variables, **classe** is the target variable.

```
file_path1 <- "./data/training.csv"

if(!file.exists(file_path1)){
  dir.create("./data")
```

```
  url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
  download.file(url, file_path1)
}


data <- read_csv("./data/training.csv")

head(data)
```

```
# A tibble: 6 x 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
# 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>, ...
```

Data Wrangling

Before going to the modeling stage, a few unnecessary columns must be eliminated. First, the columns with over 19,000 NAs are eliminated. Second, factors were created from the **classe** column. Third, train and test data are separated from the original data. Finally, a few unnecessary columns have been eliminated, including timestamp and id.

```
remove_col <- colSums(is.na(data)) > 19000
data <- data[,!remove_col]
data <- data %>%
  select(-1:-7)

data$classe <- as.factor(data$classe)

initial_split <- initial_split(data, prop = 0.8)
train <- training(initial_split)
test <- testing(initial_split)
rm(initial_split)
```

Descriptive Statistics

Below is a statistical summary of the various test data variables.

```
summary(test)
```

```
  roll_belt        pitch_belt          yaw_belt       total_accel_belt
Min.   :-28.30   Min.   :-55.8000   Min.   :-179.00   Min.   : 0.00
1st Qu.:  1.11   1st Qu.:  1.9400   1st Qu.: -88.20   1st Qu.: 3.00
Median :115.00   Median :  5.3600   Median : -11.30   Median :17.00
Mean   : 65.32   Mean   :  0.6202   Mean   : -11.39   Mean   :11.43
3rd Qu.:123.00   3rd Qu.: 15.8000   3rd Qu.:   3.85   3rd Qu.:18.00
Max.   :162.00   Max.   : 58.0000   Max.   : 179.00   Max.   :29.00
 gyros_belt_x        gyros_belt_y       gyros_belt_z      accel_belt_x
Min.   :-1.000000   Min.   :-0.51000   Min.   :-1.3500   Min.   :-68.000
1st Qu.:-0.030000   1st Qu.: 0.00000   1st Qu.:-0.2000   1st Qu.:-21.000
Median : 0.030000   Median : 0.02000   Median :-0.1100   Median :-15.000
Mean   :-0.005124   Mean   : 0.03884   Mean   :-0.1356   Mean   : -5.895
3rd Qu.: 0.110000   3rd Qu.: 0.11000   3rd Qu.:-0.0200   3rd Qu.: -5.000
Max.   : 2.020000   Max.   : 0.63000   Max.   : 1.4100   Max.   : 74.000
 accel_belt_y     accel_belt_z     magnet_belt_x    magnet_belt_y
Min.   :-69.00   Min.   :-275.00   Min.   :-52.00   Min.   :366.0
1st Qu.:  3.00   1st Qu.:-162.00   1st Qu.:  9.00   1st Qu.:581.0
Median : 38.00   Median :-154.00   Median : 34.00   Median :601.0
Mean   : 30.59   Mean   : -74.13   Mean   : 55.28   Mean   :593.6
3rd Qu.: 61.00   3rd Qu.:  27.00   3rd Qu.: 58.00   3rd Qu.:610.0
Max.   :107.00   Max.   : 105.00   Max.   :485.00   Max.   :665.0
magnet_belt_z       roll_arm         pitch_arm         yaw_arm
Min.   :-621.0   Min.   :-180.00   Min.   :-87.90   Min.   :-180.000
1st Qu.:-375.0   1st Qu.: -33.50   1st Qu.:-25.30   1st Qu.: -44.200
Median :-320.0   Median :   0.00   Median :  0.00   Median :   0.000
Mean   :-346.2   Mean   :  16.14   Mean   : -4.64   Mean   :  -1.268
3rd Qu.:-306.0   3rd Qu.:  76.40   3rd Qu.: 11.20   3rd Qu.:  44.700
Max.   : 266.0   Max.   : 178.00   Max.   : 88.20   Max.   : 179.000
total_accel_arm  gyros_arm_x        gyros_arm_y       gyros_arm_z
Min.   : 1.0    Min.   :-6.37000   Min.   :-3.2000   Min.   :-2.1300
1st Qu.:17.0    1st Qu.:-1.40000   1st Qu.:-0.7900   1st Qu.:-0.0700
Median :26.0    Median : 0.05000   Median :-0.2100   Median : 0.2300
Mean   :25.4    Mean   : 0.01309   Mean   :-0.2373   Mean   : 0.2672
3rd Qu.:33.0    3rd Qu.: 1.54000   3rd Qu.: 0.1800   3rd Qu.: 0.7200
Max.   :64.0    Max.   : 4.62000   Max.   : 2.7900   Max.   : 2.6600
 accel_arm_x       accel_arm_y       accel_arm_z       magnet_arm_x
Min.   :-336.00   Min.   :-302.00   Min.   :-608.00   Min.   :-579.0
1st Qu.:-242.00   1st Qu.: -53.00   1st Qu.:-140.00   1st Qu.:-302.0
Median : -43.00   Median :  13.00   Median : -44.00   Median : 290.0
Mean   : -61.51   Mean   :  31.76   Mean   : -69.35   Mean   : 191.2
3rd Qu.:  82.00   3rd Qu.: 136.00   3rd Qu.:  27.00   3rd Qu.: 638.0
Max.   : 434.00   Max.   : 296.00   Max.   : 242.00   Max.   : 782.0
 magnet_arm_y     magnet_arm_z    roll_dumbbell     pitch_dumbbell
Min.   :-372.0   Min.   :-597.0   Min.   :-152.83   Min.   :-130.02
1st Qu.:  -5.0   1st Qu.: 146.0   1st Qu.: -16.61   1st Qu.: -39.97
Median : 204.0   Median : 446.0   Median :  48.49   Median : -20.09
Mean   : 158.3   Mean   : 309.5   Mean   :  24.09   Mean   : -10.25
3rd Qu.: 322.0   3rd Qu.: 546.0   3rd Qu.:  68.24   3rd Qu.:  18.52
Max.   : 582.0   Max.   : 687.0   Max.   : 152.08   Max.   : 137.03
 yaw_dumbbell      total_accel_dumbbell gyros_dumbbell_x  gyros_dumbbell_y
Min.   :-144.343   Min.   : 0.0         Min.   :-1.9900   Min.   :-2.01000
1st Qu.: -77.216   1st Qu.: 4.0         1st Qu.:-0.0300   1st Qu.:-0.16000
Median :   0.000   Median :10.0         Median : 0.1300   Median : 0.05000
Mean   :   3.454   Mean   :13.7         Mean   : 0.1746   Mean   : 0.04668
```

```
 3rd Qu.:  81.752   3rd Qu.:20.0         3rd Qu.: 0.3700   3rd Qu.: 0.21000
 Max.   : 154.952   Max.   :39.0         Max.   : 1.9600   Max.   : 4.37000
 gyros_dumbbell_z  accel_dumbbell_x  accel_dumbbell_y  accel_dumbbell_z
 Min.   :-1.7700   Min.   :-237.00   Min.   :-168.00   Min.   :-272.00
 1st Qu.:-0.3100   1st Qu.: -51.00   1st Qu.:  -8.00   1st Qu.:-141.00
 Median :-0.1300   Median :  -8.00   Median :  43.00   Median :   0.00
 Mean   :-0.1426   Mean   : -27.67   Mean   :  53.56   Mean   : -37.53
 3rd Qu.: 0.0300   3rd Qu.:  11.00   3rd Qu.: 114.00   3rd Qu.:  39.00
 Max.   : 1.6700   Max.   : 219.00   Max.   : 299.00   Max.   : 318.00
 magnet_dumbbell_x magnet_dumbbell_y magnet_dumbbell_z  roll_forearm
 Min.   :-639.0    Min.   :-744.0    Min.   :-245.00   Min.   :-180.00
 1st Qu.:-534.0    1st Qu.: 234.0    1st Qu.: -48.00   1st Qu.:   0.00
 Median :-478.0    Median : 313.0    Median :  10.00   Median :  21.10
 Mean   :-323.3    Mean   : 222.8    Mean   :  43.91   Mean   :  34.78
 3rd Qu.:-284.0    3rd Qu.: 391.0    3rd Qu.:  95.00   3rd Qu.: 140.00
 Max.   : 583.0    Max.   : 631.0    Max.   : 447.00   Max.   : 180.00
 pitch_forearm     yaw_forearm      total_accel_forearm gyros_forearm_x
 Min.   :-72.10   Min.   :-180.00   Min.   : 1.00       Min.   :-3.3600
 1st Qu.:  0.00   1st Qu.: -70.00   1st Qu.:29.00       1st Qu.:-0.2200
 Median :  9.82   Median :   0.00   Median :35.00       Median : 0.0300
 Mean   : 10.65   Mean   :  18.74   Mean   :34.51       Mean   : 0.1496
 3rd Qu.: 28.00   3rd Qu.: 110.00   3rd Qu.:41.00       3rd Qu.: 0.5300
 Max.   : 87.90   Max.   : 180.00   Max.   :79.00       Max.   : 3.5200
 gyros_forearm_y    gyros_forearm_z   accel_forearm_x   accel_forearm_y
 Min.   :-6.52000   Min.   :-3.0400   Min.   :-498.00   Min.   :-595.0
 1st Qu.:-1.48000   1st Qu.:-0.2000   1st Qu.:-178.00   1st Qu.:  39.0
 Median : 0.03000   Median : 0.0700   Median : -56.00   Median : 194.0
 Mean   : 0.07435   Mean   : 0.1417   Mean   : -61.66   Mean   : 157.6
 3rd Qu.: 1.65000   3rd Qu.: 0.4900   3rd Qu.:  76.00   3rd Qu.: 309.0
 Max.   : 6.13000   Max.   : 4.0400   Max.   : 365.00   Max.   : 583.0
 accel_forearm_z  magnet_forearm_x magnet_forearm_y magnet_forearm_z classe
 Min.   :-446.0   Min.   :-1270    Min.   :-892.0   Min.   :-966     A:1128
 1st Qu.:-182.0   1st Qu.: -616    1st Qu.: -11.0   1st Qu.: 192     B: 767
 Median : -40.0   Median : -362    Median : 591.0   Median : 516     C: 691
 Mean   : -56.7   Mean   : -309    Mean   : 371.1   Mean   : 400     D: 624
 3rd Qu.:  26.0   3rd Qu.:  -72    3rd Qu.: 739.0   3rd Qu.: 655     E: 715
 Max.   : 285.0   Max.   :  672    Max.   :1450.0   Max.   :1050
```

Multinomial Logistic Regression Model

```
model_multilog <- multinom(classe~., data = train)
```

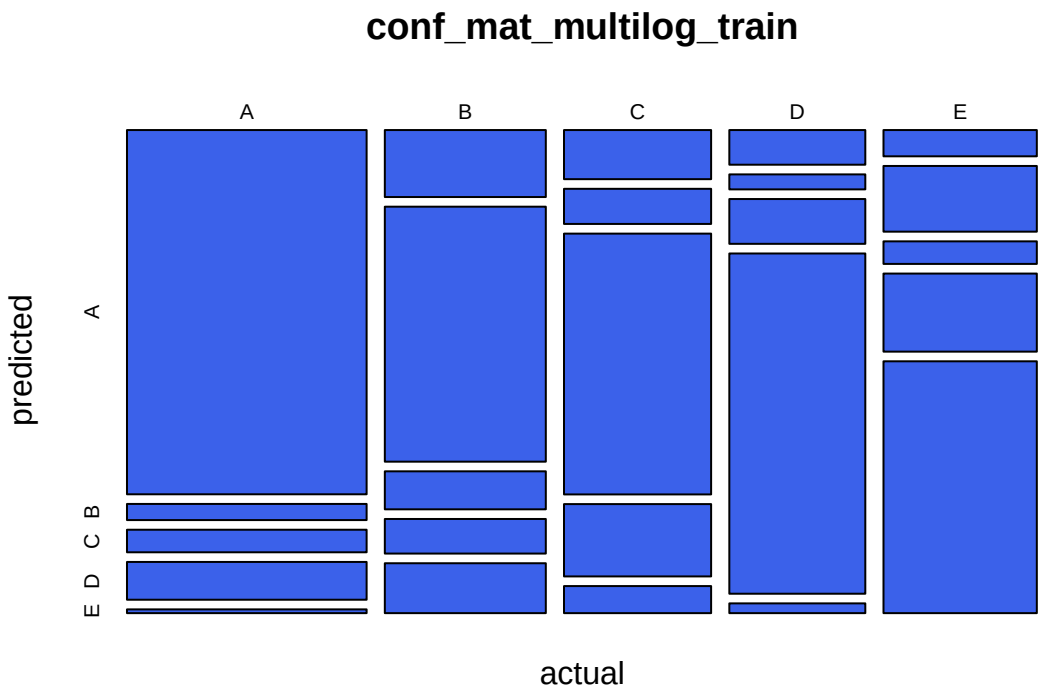
```
# weights:  270 (212 variable)
initial  value 25263.346911
iter  10 value 20053.396994
iter  20 value 17556.434839
iter  30 value 16355.170335
iter  40 value 15259.227605
iter  50 value 14687.783782
iter  60 value 14263.676758
iter  70 value 14004.687380
iter  80 value 13853.921122
```

```
iter  90 value 13749.382175
iter 100 value 13637.021377
final  value 13637.021377
stopped after 100 iterations
```

```
train <- train %>%
  mutate(pred_multilog = predict(model_multilog, newdata = train))

test <- test %>%
  mutate(pred_multilog = predict(model_multilog, newdata = test))
```

Accuracy Measures

```
conf_mat_multilog_train <- table(actual = train$classe, predicted = train$pred_multilog)

conf_mat_multilog_test <- table(actual = test$classe, predicted = test$pred_multilog)

plot(conf_mat_multilog_train, color = "#3B61EA")
```

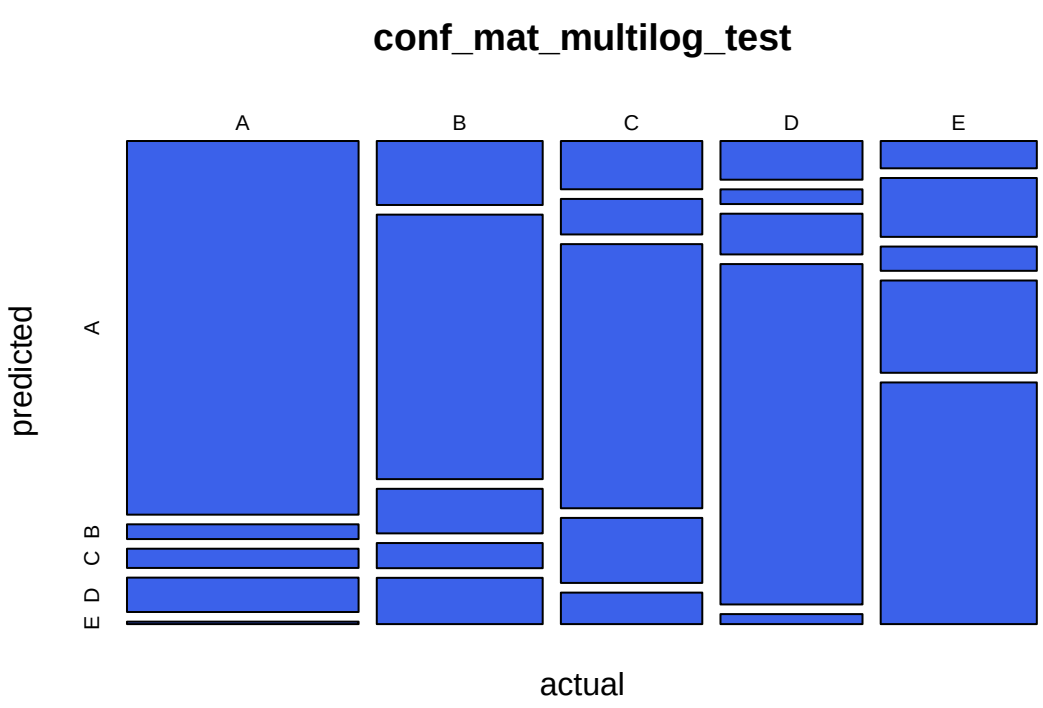



```
confusionMatrix(conf_mat_multilog_train)
```

```
Confusion Matrix and Statistics
```

```
      predicted
actual    A    B    C    D    E
     A 3683  163  228  382   38
     B  455 1732  258  235  339
     C  305  218 1619  449  168
     D  199   86  257 1952   56
     E  170  425  146  505 1629

Overall Statistics

               Accuracy : 0.6762
                 95% CI : (0.6689, 0.6836)
    No Information Rate : 0.3066
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5899

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.7654   0.6601   0.6455   0.5541   0.7305
Specificity            0.9255   0.9016   0.9136   0.9509   0.9075
Pos Pred Value         0.8195   0.5737   0.5868   0.7655   0.5666
Neg Pred Value         0.8992   0.9296   0.9313   0.8805   0.9531
Prevalence             0.3066   0.1672   0.1598   0.2244   0.1421
Detection Rate         0.2346   0.1103   0.1031   0.1244   0.1038
Detection Prevalence   0.2863   0.1923   0.1758   0.1625   0.1832
Balanced Accuracy      0.8454   0.7808   0.7795   0.7525   0.8190
```

```r
plot(conf_mat_multilog_test, color = "#3B61EA")
```




```
confusionMatrix(conf_mat_multilog_test)
```

```
Confusion Matrix and Statistics

          predicted
actual      A   B   C   D   E
     A    913  36  47  84   6
     B    112 463  78  44  81
     C     72  53 394  97  47
     D     58  22  61 510  15
     E     45  97  40 152 398

Overall Statistics

               Accuracy : 0.6823
                 95% CI : (0.6675, 0.6968)
    No Information Rate : 0.3057
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5981

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:
```

```
                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.7608   0.6900   0.6355   0.5750   0.7276
Specificity            0.9365   0.9032   0.9186   0.9487   0.9011
Pos Pred Value         0.8407   0.5951   0.5943   0.7658   0.5437
Neg Pred Value         0.8989   0.9339   0.9307   0.8843   0.9533
Prevalence             0.3057   0.1710   0.1580   0.2260   0.1394
Detection Rate         0.2326   0.1180   0.1004   0.1299   0.1014
Detection Prevalence   0.2767   0.1982   0.1689   0.1697   0.1865
Balanced Accuracy      0.8487   0.7966   0.7770   0.7618   0.8144
```

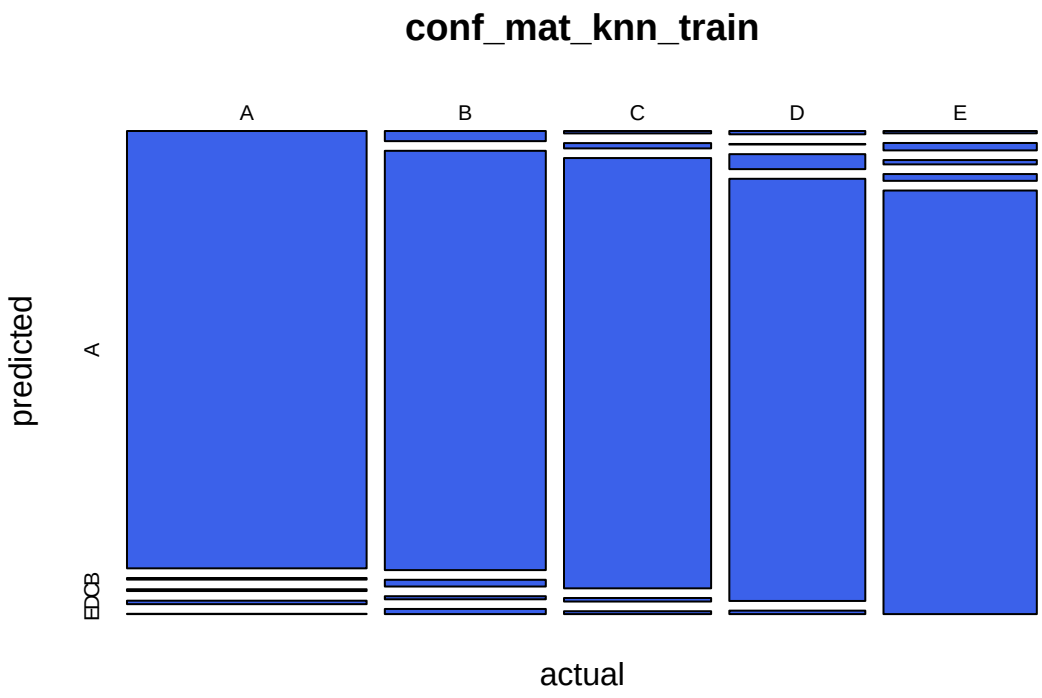
KNN

```
model_knn <- train(classe~., data = train, method = "knn")
model_knn
```

```
k-Nearest Neighbors

15697 samples
   53 predictor
    5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 15697, 15697, 15697, 15697, 15697, 15697, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.8837723  0.8528095
  7  0.8679763  0.8327818
  9  0.8556265  0.8171290

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
```

Accuracy measures

```
train$pred_knn <- predict(model_knn, newdata = train)

conf_mat_knn_train <- table(actual = train$classe,
                            predicted = train$pred_knn)

plot(conf_mat_knn_train, color = "#3B61EA")
```

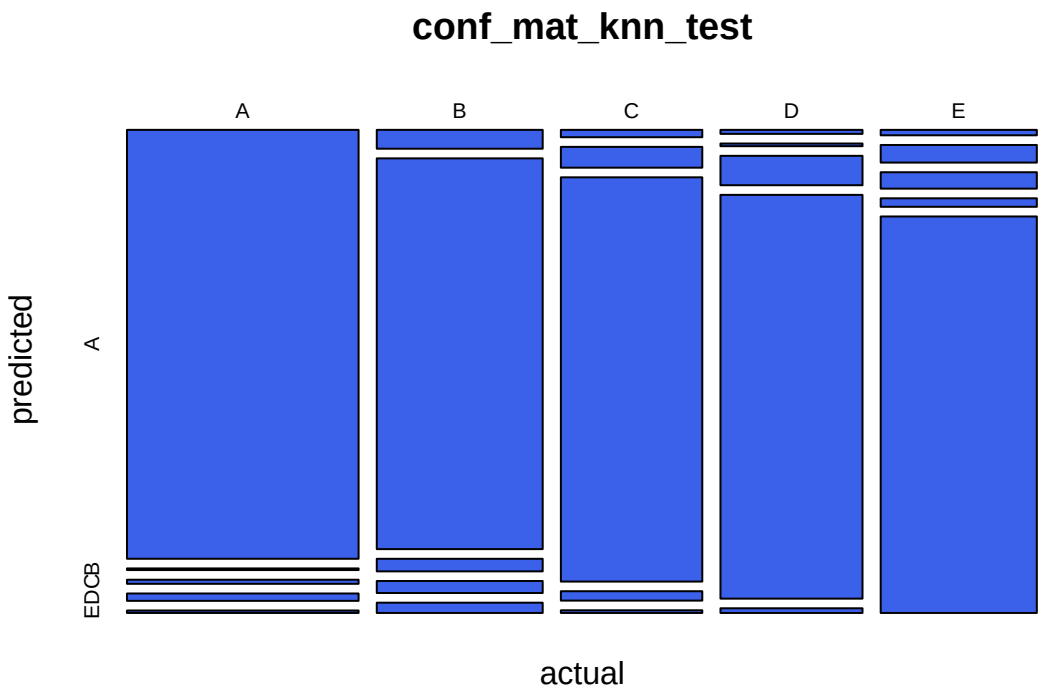



```
confusionMatrix(conf_mat_knn_train)
```

```
Confusion Matrix and Statistics

          predicted
actual    A    B    C    D    E
     A 4422   16   18   35    3
     B   69 2848   46   21   35
     C   15   33 2671   22   18
     D   19    3   86 2422   20
     E   15   48   28   44 2740

Overall Statistics

               Accuracy : 0.9622
                 95% CI : (0.9591, 0.9651)
    No Information Rate : 0.2892
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9521

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:
```

```
                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9740   0.9661   0.9375   0.9520   0.9730
Specificity            0.9935   0.9866   0.9932   0.9903   0.9895
Pos Pred Value         0.9840   0.9434   0.9681   0.9498   0.9530
Neg Pred Value         0.9895   0.9921   0.9862   0.9907   0.9941
Prevalence             0.2892   0.1878   0.1815   0.1621   0.1794
Detection Rate         0.2817   0.1814   0.1702   0.1543   0.1746
Detection Prevalence   0.2863   0.1923   0.1758   0.1625   0.1832
Balanced Accuracy      0.9838   0.9763   0.9653   0.9712   0.9813
```

```r
test$pred_knn <- predict(model_knn, newdata = test)

conf_mat_knn_test <- table(actual = test$classe,
                           predicted = test$pred_knn)

plot(conf_mat_knn_test, color = "#3B61EA")
```



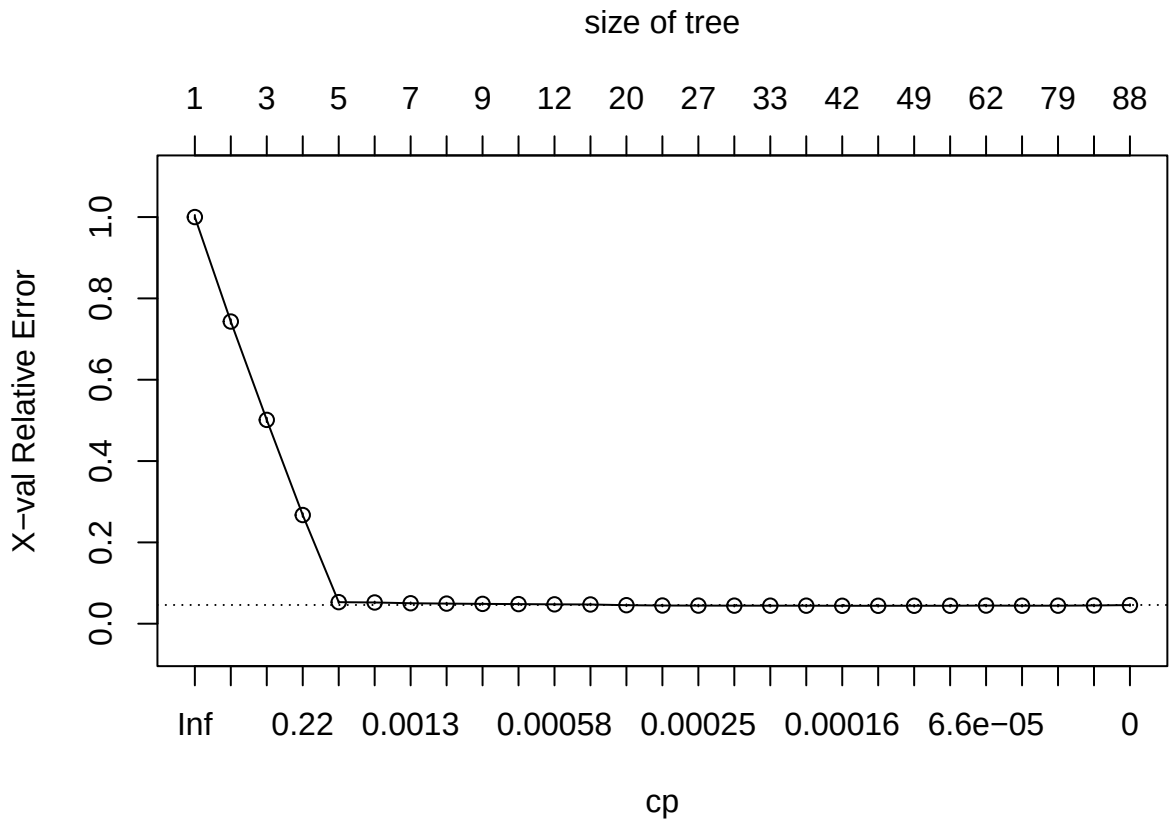

```r
confusionMatrix(conf_mat_knn_test)
```

```
Confusion Matrix and Statistics

      predicted
actual    A    B    C    D    E
     A 1048    4   10   18    6
     B   33  684   22   21   18
```

```
     C   11   31  603   14    4
     D    6    4   44  605    7
     E    9   29   27   14  653

Overall Statistics

               Accuracy : 0.9154
                 95% CI : (0.9063, 0.9239)
    No Information Rate : 0.282
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8932

 Mcnemar's Test P-Value : 5.123e-13

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9467   0.9096   0.8541   0.9003   0.9491
Specificity            0.9865   0.9704   0.9814   0.9812   0.9756
Pos Pred Value         0.9650   0.8792   0.9095   0.9084   0.8921
Neg Pred Value         0.9792   0.9784   0.9684   0.9794   0.9890
Prevalence             0.2820   0.1916   0.1799   0.1712   0.1753
Detection Rate         0.2670   0.1743   0.1536   0.1541   0.1664
Detection Prevalence   0.2767   0.1982   0.1689   0.1697   0.1865
Balanced Accuracy      0.9666   0.9400   0.9177   0.9408   0.9624
```

Decision Trees

```
model_dc <- rpart(classe~., data = train, method = "class",
                  control = rpart.control(cp = 0))

plotcp(model_dc)
```




```
printcp(model_dc)
```

```
Classification tree:
rpart(formula = classe ~ ., data = train, method = "class", control = rpart.control(cp = 0))

Variables actually used in tree construction:
 [1] accel_arm_x       accel_arm_z       accel_belt_z      accel_dumbbell_z
 [5] accel_forearm_y   gyros_arm_x       gyros_arm_y       gyros_belt_x
 [9] gyros_belt_y      gyros_belt_z      gyros_dumbbell_x  gyros_dumbbell_y
[13] gyros_forearm_x   gyros_forearm_y   gyros_forearm_z   magnet_arm_x
[17] magnet_arm_y      magnet_belt_x     magnet_belt_y     magnet_dumbbell_y
[21] magnet_dumbbell_z magnet_forearm_x  magnet_forearm_z  pitch_arm
[25] pitch_belt        pitch_forearm     pred_knn          pred_multilog
[29] roll_arm          roll_belt         roll_dumbbell     roll_forearm
[33] yaw_belt

Root node error: 11203/15697 = 0.7137

n= 15697

           CP nsplit rel error   xerror      xstd
1  2.5690e-01      0  1.000000 1.000000 0.0050552
2  2.4190e-01      1  0.743105 0.743105 0.0055814
3  2.3396e-01      2  0.501205 0.501205 0.0053605
```

```
4  2.1423e-01      3  0.267250 0.267250 0.0043938
5  1.6960e-03      4  0.053022 0.053022 0.0021339
6  1.6067e-03      5  0.051326 0.052218 0.0021183
7  1.0711e-03      6  0.049719 0.050344 0.0020814
8  9.8188e-04      7  0.048648 0.049451 0.0020636
9  7.1409e-04      8  0.047666 0.048737 0.0020492
10 6.2483e-04     10  0.046238 0.048201 0.0020383
11 5.3557e-04     11  0.045613 0.047666 0.0020273
12 5.1326e-04     14  0.044006 0.047309 0.0020200
13 2.6779e-04     19  0.041150 0.045524 0.0019828
14 2.5291e-04     20  0.040882 0.044720 0.0019658
15 2.3803e-04     26  0.039364 0.044542 0.0019620
16 2.0828e-04     29  0.038650 0.044452 0.0019601
17 1.7852e-04     32  0.038026 0.044363 0.0019582
18 1.6067e-04     36  0.037311 0.044363 0.0019582
19 1.5621e-04     41  0.036508 0.044095 0.0019525
20 1.4877e-04     45  0.035883 0.044095 0.0019525
21 1.1158e-04     48  0.035437 0.044185 0.0019544
22 8.9262e-05     52  0.034991 0.044185 0.0019544
23 4.8688e-05     61  0.034187 0.044631 0.0019639
24 4.4631e-05     72  0.033652 0.044274 0.0019563
25 2.9754e-05     78  0.033384 0.044363 0.0019582
26 1.4877e-05     81  0.033295 0.044899 0.0019696
27 0.0000e+00     87  0.033205 0.045791 0.0019884
```

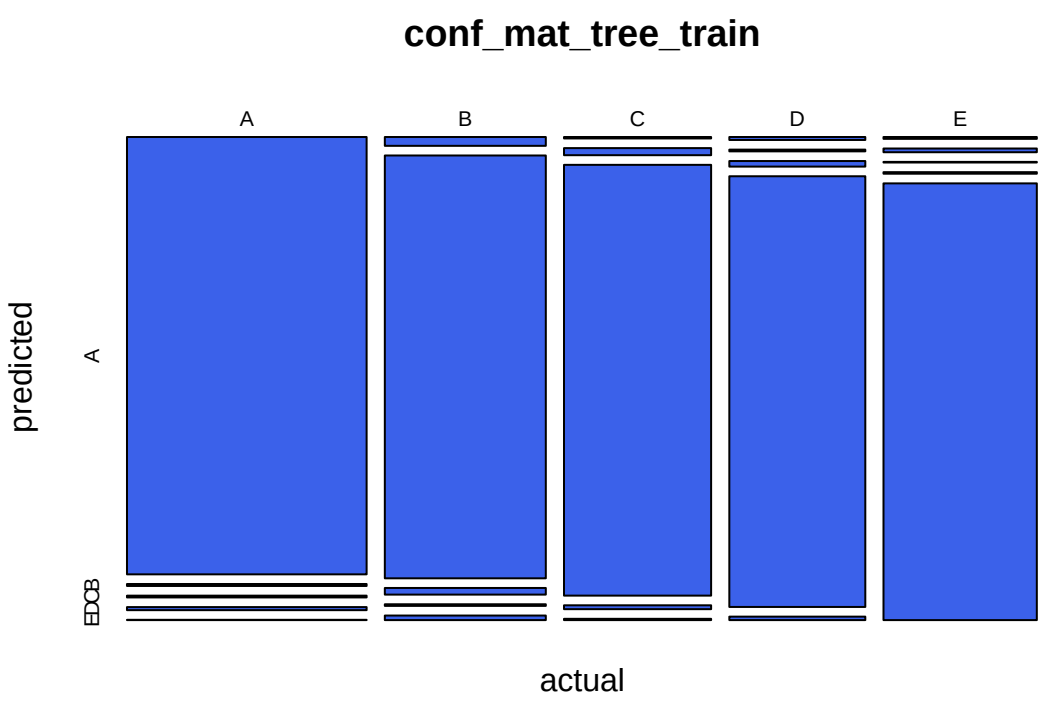
Pruning

Pruning a decision tree based on complexity parameter allows to extract better results from a decision tree model.

```
model_dc_pruned <- prune(model_dc, cp = 2.2799e-04)
rm(model_dc)
```

Accuracy Measures

```
train$pred_dc <- predict(model_dc_pruned, newdata = train, type = "class")

conf_mat_tree_train <- table(actual = train$classe,
                             predicted = train$pred_dc)

plot(conf_mat_tree_train, color = "#3B61EA")
```

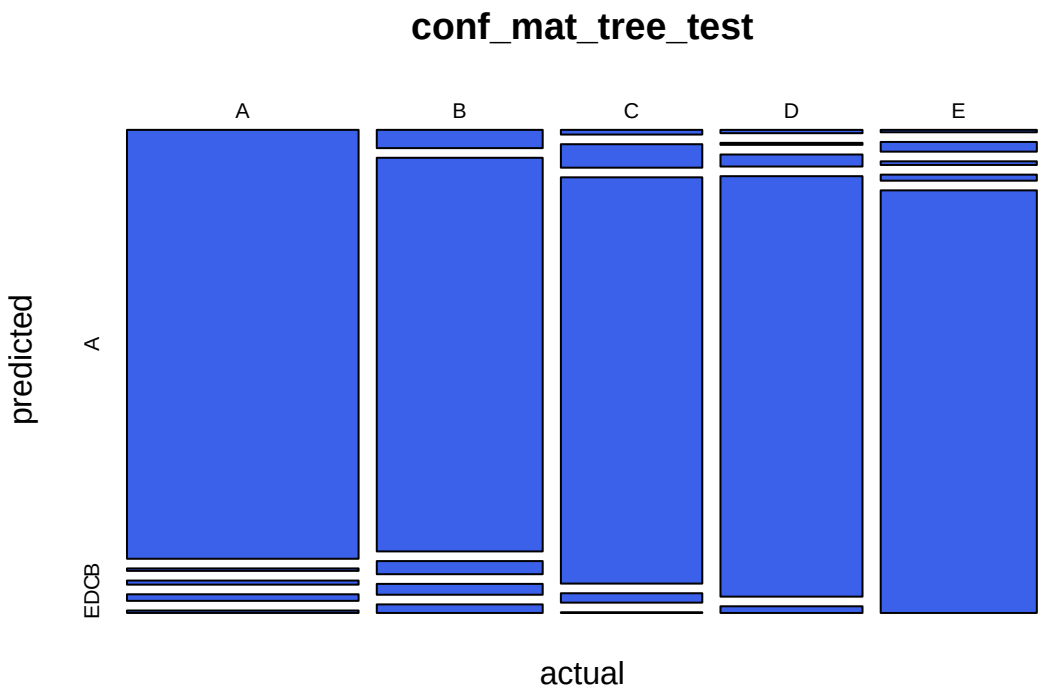



```
confusionMatrix(conf_mat_tree_train)
```

```
Confusion Matrix and Statistics

          predicted
actual       A    B    C    D    E
     A    4422   20   18   31    3
     B      60 2872   45   11   31
     C       9   44 2674   24    8
     D      17   10   32 2471   20
     E      12   24    4   10 2825

Overall Statistics

                  Accuracy : 0.9724
                    95% CI : (0.9697, 0.9749)
    No Information Rate : 0.288
    P-Value [Acc > NIR] : < 2.2e-16

                     Kappa : 0.9651

 Mcnemar's Test P-Value : 2.302e-05

Statistics by Class:
```

```
                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9783   0.9670   0.9643   0.9702   0.9785
Specificity            0.9936   0.9884   0.9934   0.9940   0.9961
Pos Pred Value         0.9840   0.9513   0.9692   0.9690   0.9826
Neg Pred Value         0.9913   0.9923   0.9923   0.9942   0.9952
Prevalence             0.2880   0.1892   0.1767   0.1623   0.1839
Detection Rate         0.2817   0.1830   0.1704   0.1574   0.1800
Detection Prevalence   0.2863   0.1923   0.1758   0.1625   0.1832
Balanced Accuracy      0.9859   0.9777   0.9789   0.9821   0.9873
```

```r
test$pred_dc <- predict(model_dc_pruned, newdata = test, type = "class")

conf_mat_tree_test <- table(actual = test$classe,
                            predicted = test$pred_dc)

plot(conf_mat_tree_test, color = "#3B61EA")
```




```r
confusionMatrix(conf_mat_tree_test)
```

```
Confusion Matrix and Statistics

      predicted
actual    A    B    C    D    E
     A 1048    6   10   16    6
     B   32  689   23   19   15
```

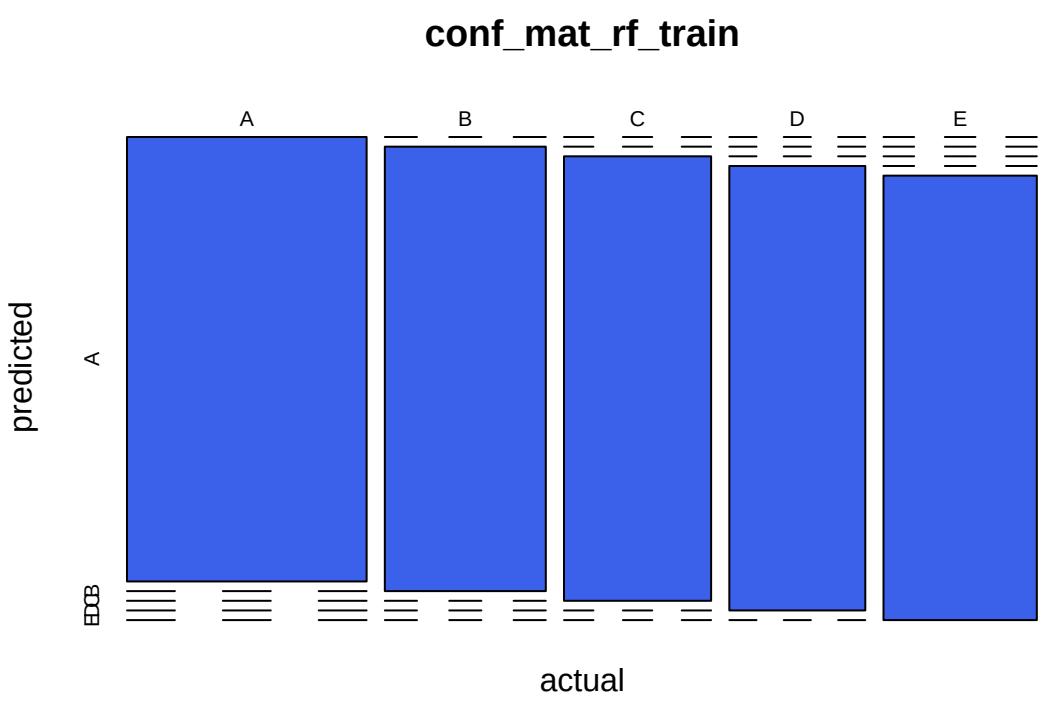
```
     C    7   35  606   14    1
     D    5    3   18  630   10
     E    4   16    6   10  696

Overall Statistics

               Accuracy : 0.9348
                 95% CI : (0.9266, 0.9423)
    No Information Rate : 0.2792
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9176

 Mcnemar's Test P-Value : 5.618e-06

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9562   0.9199   0.9140   0.9144   0.9560
Specificity            0.9866   0.9720   0.9825   0.9889   0.9887
Pos Pred Value         0.9650   0.8856   0.9140   0.9459   0.9508
Neg Pred Value         0.9831   0.9809   0.9825   0.9819   0.9900
Prevalence             0.2792   0.1908   0.1689   0.1755   0.1855
Detection Rate         0.2670   0.1755   0.1544   0.1605   0.1773
Detection Prevalence   0.2767   0.1982   0.1689   0.1697   0.1865
Balanced Accuracy      0.9714   0.9459   0.9483   0.9516   0.9724
```

Ensemble Method: Random Forest

```
model_rf <- randomForest(formula = classe~., data = train, ntree = 500)
```

Accuracy Measures

```
train$pred_rf <- predict(model_rf, newdata = train)
conf_mat_rf_train <- table(actual = train$classe, predicted = train$pred_rf)

plot(conf_mat_rf_train, color = "#3B61EA")
```

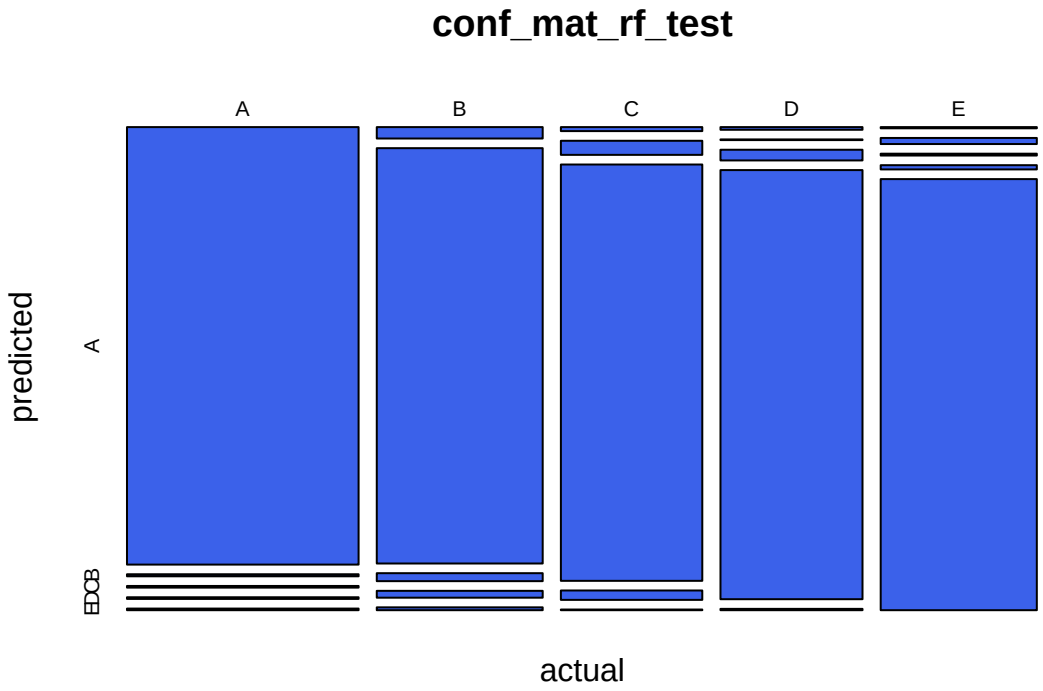



```
confusionMatrix(conf_mat_rf_train)
```

```
## Confusion Matrix and Statistics
##
##       predicted
## actual    A    B    C    D    E
##      A 4494    0    0    0    0
##      B    0 3019    0    0    0
##      C    0    0 2759    0    0
##      D    0    0    0 2550    0
##      E    0    0    0    0 2875
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2863
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2863   0.1923   0.1758   0.1625   0.1832
## Detection Rate         0.2863   0.1923   0.1758   0.1625   0.1832
## Detection Prevalence   0.2863   0.1923   0.1758   0.1625   0.1832
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```r
test$pred_rf <- predict(model_rf, newdata = test)
conf_mat_rf_test <- table(actual = test$classe, predicted = test$pred_rf)

plot(conf_mat_rf_test, color = "#3B61EA")
```




```r
confusionMatrix(conf_mat_rf_test)
```

```
## Confusion Matrix and Statistics
##
##        predicted
## actual    A    B    C    D    E
##      A 1069    5    4    4    4
##      B   20  727   14   12    5
##      C    6   21  621   14    1
##      D    4    1   16  643    2
```

```
##      E    2   10    3    7  710
##
## Overall Statistics
##
##                Accuracy : 0.9605
##                  95% CI : (0.9539, 0.9664)
##     No Information Rate : 0.2805
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9501
##
##  Mcnemar's Test P-Value : 0.003294
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9709   0.9516   0.9438   0.9456   0.9834
## Specificity            0.9940   0.9839   0.9871   0.9929   0.9931
## Pos Pred Value         0.9843   0.9344   0.9367   0.9655   0.9699
## Neg Pred Value         0.9887   0.9882   0.9887   0.9886   0.9962
## Prevalence             0.2805   0.1946   0.1676   0.1732   0.1839
## Detection Rate         0.2724   0.1852   0.1582   0.1638   0.1809
## Detection Prevalence   0.2767   0.1982   0.1689   0.1697   0.1865
## Balanced Accuracy      0.9825   0.9677   0.9655   0.9693   0.9883
```

Conclusion

This study demonstrates the enormous potential of wearable technology in the medical field, particularly when paired with machine learning and sophisticated data analytics. This study shows that information from wearable sensors may accurately predict how exercises are completed by classifying human activity, especially the quality of exercise form. The study shows how R-based machine learning methods, including random forests, decision trees, k-nearest neighbors, and multinomial logistic regression, can accurately distinguish between workout movements that are executed correctly and incorrectly.

The random forest classifier produced the highest overall accuracy among the tested models, demonstrating its resilience while processing high-dimensional, complicated data, such accelerometer measurements from several sensors. This is in line with findings in scholarly literature that highlight the applicability of random forests for wearable data because of their capacity to manage heterogeneous feature sets and their resistance to overfitting (Rodgers et al., 2019; Wickham, 2019). The project also emphasizes the significance of feature engineering and data preprocessing, which were essential for removing duplicate or unnecessary data and improving model performance. Kuhn and Johnson (2019) confirmed this procedure in their work on feature selection.

The project's insights have important ramifications for applications in healthcare. By enabling medical personnel to precisely watch patient development and take swift action when aberrations in exercise form are identified, remote exercise quality monitoring has the potential to completely transform physical therapy and rehabilitation. By spotting unusual movement patterns that could point to the early stages of musculoskeletal problems, this method can be applied to preventive care (Bejarano et al., 2014). Additionally, wearable sensor data combined with R-based analytics presents a feasible solution for remote patient monitoring as telehealth gains traction, especially in the wake of the COVID-19 pandemic.

All things considered, this study shows how R-based analytics can greatly improve our capacity to decipher and respond to human activity data in healthcare settings when applied to wearable sensor data. Analytics-driven insights can assist therapeutic interventions and preventive measures by enabling accu-

rate and continuous tracking of patient activities, which can lead to better health outcomes and a more individualized approach to care. Neural networks may provide even more detailed insights from wearable data, thus future studies might investigate incorporating deep learning techniques for activity detection. The generalizability of these models might also be strengthened by extending this analysis to encompass a larger demographic sample and a greater range of activities, opening the door for their use in many health scenarios. As a result, this initiative establishes the foundation for applied analytics in wearable technology, highlighting its revolutionary potential in contemporary healthcare.

References

Bejarano, N. C., Ambrosini, E., Pedrocchi, A., Ferrigno, G., Monticone, M., & Ferrante, S. (2014). A novel adaptive, real-time algorithm to detect gait events from wearable sensors. IEEE transactions on neural systems and rehabilitation engineering, 23(3), 413-422.

Breiman, L. (2001). Random forests. Machine learning, 45, 5-32.

Kuhn, M. (2008). Building predictive models in R using the caret package. Journal of statistical software, 28, 1-26.

Kuhn, M. (2013). Applied predictive modeling.

Kuhn, M., & Johnson, K. (2019). Feature engineering and selection: A practical approach for predictive models. Chapman and Hall/CRC

Mukhopadhyay, S. C. (2014). Wearable sensors for human activity monitoring: A review. IEEE sensors journal, 15(3), 1321-1330.

Piwek, L., Ellis, D. A., Andrews, S., & Joinson, A. (2016). The rise of consumer health wearables: promises and barriers. PLoS medicine, 13(2), e1001953.

Rodgers, M. M., Alon, G., Pai, V. M., & Conroy, R. S. (2019). Wearable technologies for active living and rehabilitation: Current research challenges and future opportunities. Journal of rehabilitation and assistive technologies engineering, 6, 2055668319839607.

Therneau, T. M., & Atkinson, E. J. (1997). An introduction to recursive partitioning using the RPART routines (Vol. 61, p. 452). Mayo Foundation: Technical report.

Venables, W. N., & Ripley, B. D. (2013). Modern applied statistics with S-PLUS. Springer Science & Business Media.

Wang, S., Yang, J., Chen, N., Chen, X., & Zhang, Q. (2005, October). Human activity recognition with user-free accelerometers in the sensor networks. In 2005 International Conference on Neural Networks and Brain (Vol. 2, pp. 1212-1217). IEEE.

Wickham, H. (2019). Advanced r. chapman and hall/CRC.