Markdown

Aadil

2024-10-03

Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to 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. More information is available from the website here: (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Loading the Required Datasets and the dependencies:

```
library(lattice)
library(ggplot2)
library(caret)
library(kernlab)
library(rattle)
library(corrplot)
set.seed(1234)

trainingData <- read.csv("pml-training.csv")
testingData <- read.csv("pml-testing.csv")
```

The dimensions and valuable information about the nature of the columns present in the dataset can be determined using the code below:

```
dim(trainingData)
```

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

```
str(trainingData)
```

```
## 'data.frame':    19622 obs. of  160 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 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
##  $ 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_roll_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt           : chr  "" "" "" "" ...
##  $ min_roll_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt           : chr  "" "" "" "" ...
##  $ amplitude_roll_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt     : chr  "" "" "" "" ...
##  $ var_total_accel_belt   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##  $ var_accel_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ 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  "" "" "" "" ...
##  $ max_roll_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm            : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm            : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm      : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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_roll_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell       : chr  "" "" "" "" ...
##  $ min_roll_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell       : chr  "" "" "" "" ...
##  $ amplitude_roll_dumbbell: num  NA NA NA NA NA NA NA NA NA NA ...
##    [list output truncated]
```

```
dim(testingData)
```

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

```
str(testingData)
```

```
## 'data.frame':    20 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : chr  "pedro" "jeremy" "jeremy" "adelmo" ...
##  $ raw_timestamp_part_1    : int  1323095002 1322673067 1322673075 1322832789 1322489635 1322673149 1
##  $ raw_timestamp_part_2    : int  868349 778725 342967 560311 814776 510661 766645 54671 916313 38428
```

```
##  $ cvtd_timestamp         : chr  "05/12/2011 14:23" "30/11/2011 17:11" "30/11/2011 17:11" "02/12/20
##  $ new_window             : chr  "no" "no" "no" "no" ...
##  $ num_window             : int  74 431 439 194 235 504 485 440 323 664 ...
##  $ roll_belt              : num  123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.93 114 ...
##  $ pitch_belt             : num  27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.72 22.4 ...
##  $ yaw_belt               : num  -4.75 -88.9 -88.5 162 -88.6 -87.7 -87.3 -88.5 -93.7 -13.1 ...
##  $ total_accel_belt       : int  20 4 5 17 3 4 4 4 4 18 ...
##  $ kurtosis_roll_belt     : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt    : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt      : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt     : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1   : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_belt      : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt          : logi  NA NA NA NA NA NA ...
##  $ max_picth_belt         : logi  NA NA NA NA NA NA ...
##  $ max_yaw_belt           : logi  NA NA NA NA NA NA ...
##  $ min_roll_belt          : logi  NA NA NA NA NA NA ...
##  $ min_pitch_belt         : logi  NA NA NA NA NA NA ...
##  $ min_yaw_belt           : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_belt    : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt   : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt     : logi  NA NA NA NA NA NA ...
##  $ var_total_accel_belt   : logi  NA NA NA NA NA NA ...
##  $ avg_roll_belt          : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_belt       : logi  NA NA NA NA NA NA ...
##  $ var_roll_belt          : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_belt         : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_belt      : logi  NA NA NA NA NA NA ...
##  $ var_pitch_belt         : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_belt           : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_belt        : logi  NA NA NA NA NA NA ...
##  $ var_yaw_belt           : logi  NA NA NA NA NA NA ...
##  $ gyros_belt_x           : num  -0.5 -0.06 0.05 0.11 0.03 0.1 -0.06 -0.18 0.1 0.14 ...
##  $ gyros_belt_y           : num  -0.02 -0.02 0.02 0.11 0.02 0.05 0 -0.02 0 0.11 ...
##  $ gyros_belt_z           : num  -0.46 -0.07 0.03 -0.16 0 -0.13 0 -0.03 -0.02 -0.16 ...
##  $ accel_belt_x           : int  -38 -13 1 46 -8 -11 -14 -10 -15 -25 ...
##  $ accel_belt_y           : int  69 11 -1 45 4 -16 2 -2 1 63 ...
##  $ accel_belt_z           : int  -179 39 49 -156 27 38 35 42 32 -158 ...
##  $ magnet_belt_x          : int  -13 43 29 169 33 31 50 39 -6 10 ...
##  $ magnet_belt_y          : int  581 636 631 608 566 638 622 635 600 601 ...
##  $ magnet_belt_z          : int  -382 -309 -312 -304 -418 -291 -315 -305 -302 -330 ...
##  $ roll_arm               : num  40.7 0 0 -109 76.1 0 0 0 -137 -82.4 ...
##  $ pitch_arm              : num  -27.8 0 0 55 2.76 0 0 0 11.2 -63.8 ...
##  $ yaw_arm                : num  178 0 0 -142 102 0 0 0 -167 -75.3 ...
##  $ total_accel_arm        : int  10 38 44 25 29 14 15 22 34 32 ...
##  $ var_accel_arm          : logi  NA NA NA NA NA NA ...
##  $ avg_roll_arm           : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_arm        : logi  NA NA NA NA NA NA ...
##  $ var_roll_arm           : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_arm          : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_arm       : logi  NA NA NA NA NA NA ...
##  $ var_pitch_arm          : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_arm            : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_arm         : logi  NA NA NA NA NA NA ...
```

```
##  $ var_yaw_arm             : logi  NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  -1.65 -1.17 2.1 0.22 -1.96 0.02 2.36 -3.71 0.03 0.26 ...
##  $ gyros_arm_y             : num  0.48 0.85 -1.36 -0.51 0.79 0.05 -1.01 1.85 -0.02 -0.5 ...
##  $ gyros_arm_z             : num  -0.18 -0.43 1.13 0.92 -0.54 -0.07 0.89 -0.69 -0.02 0.79 ...
##  $ accel_arm_x             : int  16 -290 -341 -238 -197 -26 99 -98 -287 -301 ...
##  $ accel_arm_y             : int  38 215 245 -57 200 130 79 175 111 -42 ...
##  $ accel_arm_z             : int  93 -90 -87 6 -30 -19 -67 -78 -122 -80 ...
##  $ magnet_arm_x            : int  -326 -325 -264 -173 -170 396 702 535 -367 -420 ...
##  $ magnet_arm_y            : int  385 447 474 257 275 176 15 215 335 294 ...
##  $ magnet_arm_z            : int  481 434 413 633 617 516 217 385 520 493 ...
##  $ kurtosis_roll_arm       : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm      : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm        : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_arm       : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_arm      : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_arm        : logi  NA NA NA NA NA NA ...
##  $ max_roll_arm            : logi  NA NA NA NA NA NA ...
##  $ max_picth_arm           : logi  NA NA NA NA NA NA ...
##  $ max_yaw_arm             : logi  NA NA NA NA NA NA ...
##  $ min_roll_arm            : logi  NA NA NA NA NA NA ...
##  $ min_pitch_arm           : logi  NA NA NA NA NA NA ...
##  $ min_yaw_arm             : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : logi  NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  -17.7 54.5 57.1 43.1 -101.4 ...
##  $ pitch_dumbbell          : num  25 -53.7 -51.4 -30 -53.4 ...
##  $ yaw_dumbbell            : num  126.2 -75.5 -75.2 -103.3 -14.2 ...
##  $ kurtosis_roll_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : logi  NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : logi  NA NA NA NA NA NA ...
##   [list output truncated]
```

Data Cleaning

By running 'str()', it is observed that a major portion of the dataset constitutes 'NA' values as the data is unclean. Before we proceed to perform any exploratory analysis or run any machine learning algorithm, the data has to be cleaned. This is achieved by removing columns which do not provide any data (apart from NA values)

```
trainingNA <- trainingData[,colMeans(is.na(trainingData)) < .9]
dim(trainingNA)
```

```
## [1] 19622    93
```

```
dim(trainingData)
```

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

It is observed that this one step has already removed approximately 65 columns from the training dataset. By observing the columns obtained in the previous step using the str() function, it is seen that the first 7 columns till 'num_window' only constitute the metadata and provide no useful information for the actual data. These columns are subsequently removed from the dataset.

```
trainingMeta <- trainingNA[,-c(1:7)]
dim(trainingMeta)
```

```
## [1] 19622    86
```

The 'zero-variance' variables are also subsequently removed:

```
nvz <- nearZeroVar(trainingMeta)
trainingZV <- trainingMeta[,-nvz]
dim(trainingZV)
```

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

This cleaned 'training' data set is subsequently split into a validation and a proper training case in order to apply the required machine learning algorithms. It is split in a 70-30 ratio.

```
inTrain <- createDataPartition(y=trainingZV$classe, p=0.7, list=F)
training <- trainingZV[inTrain,]
validation <- trainingZV[-inTrain,]

dim(training)
```

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

```
dim(validation)
```

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

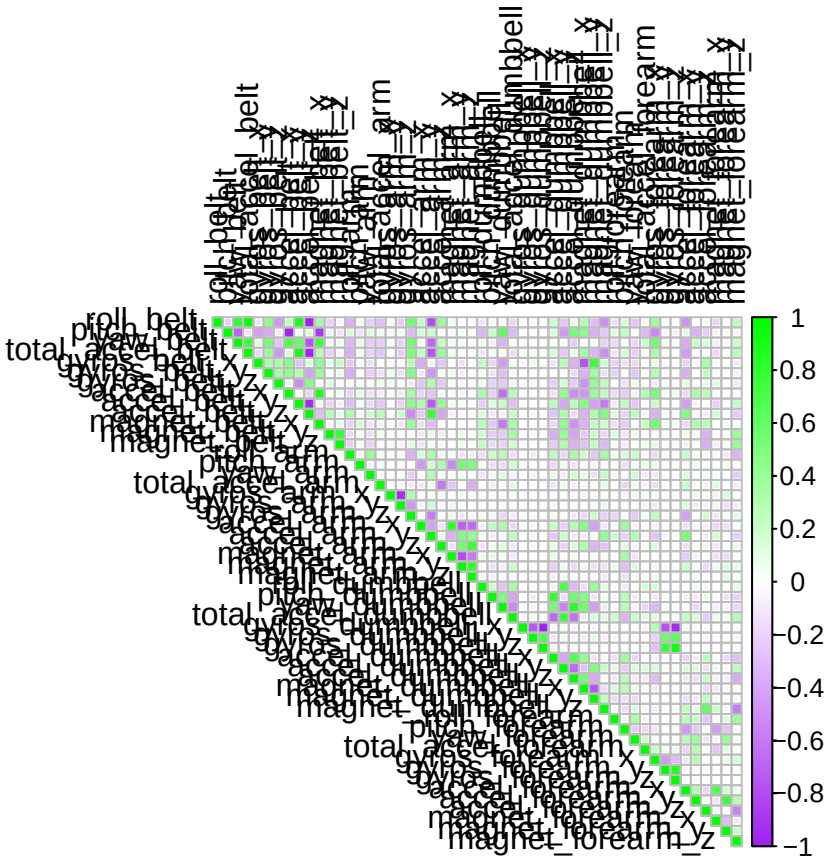
Results

In this project, we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise. (Setting up the control for 3-fold cross-validation)

```
control <- trainControl(method="cv", number=3, verboseIter=F)
```

Before implementing any known model, exploratory analysis is carried out in the form of a correlation matrix to detewrmine the relative correlation between the variables used to predict in the dataset.

```
library(caret)
corrPlot <- cor(training[, -length(names(training))])
corrplot(corrPlot, method="square", type="upper", tl.col="Black", col = colorRampPalette(c("purple", "wh
```



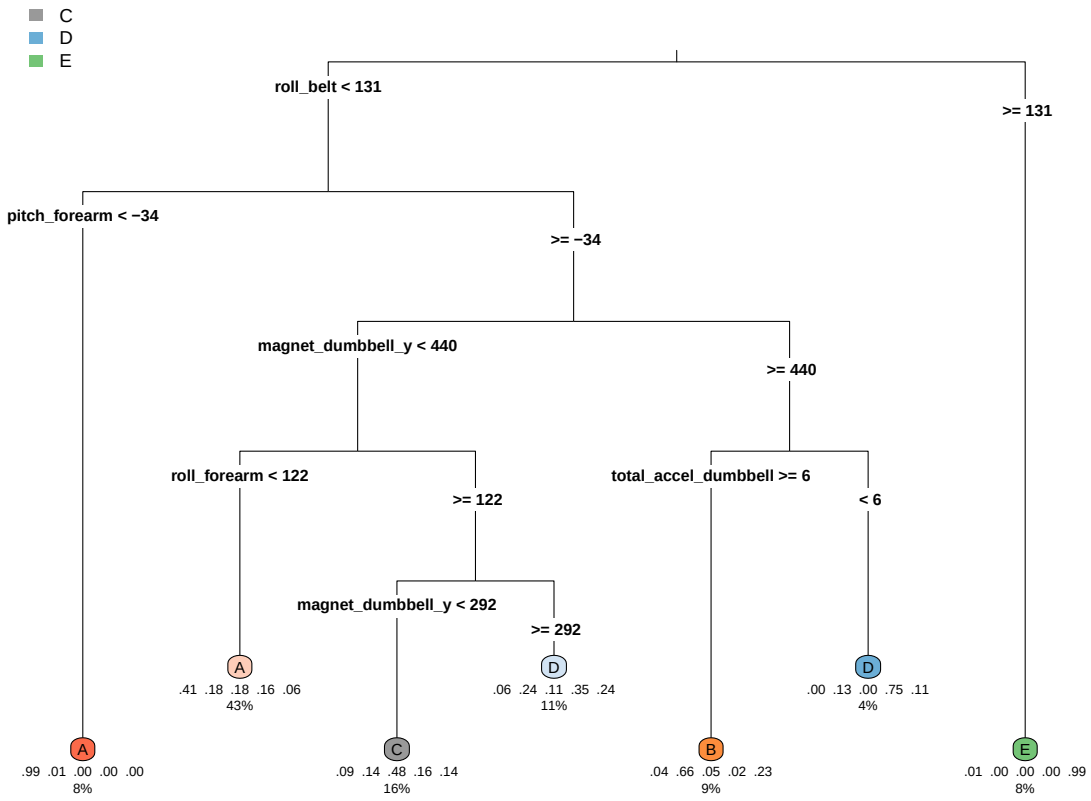
The predictions obtained between two ML algorithms; decision trees and SVM; will be compared to determine the best algorithm to make predictions of the dataset.

Decision Trees:

A decision tree is a popular machine learning algorithm used for both classification and regression tasks. It works by recursively splitting the dataset into subsets based on feature values, creating a tree-like structure where each internal node represents a feature, each branch represents a decision based on that feature, and each leaf node represents the output (a class label for classification or a value for regression). It is easy to understand, handles non-linear data and does not require data scaling or normalization. However, it is prone to over-fitting, and has an inherent bias towards dominant features.

The decision tree is plotted as shown below

```
library(rpart)
library(rpart.plot)
dec_trees <- train(classe~., data=training, method="rpart", trControl = control, tuneLength = 5)
rpart.plot(dec_trees$finalModel, type = 3, extra = 104, under = TRUE, fallen.leaves = TRUE)
```



The statistical predictions and the confusion matrix made by the decision tree on the validation data can be obtained using the code below:

```
dec_trees_pred <- predict(dec_trees, validation)
trees_pred <- confusionMatrix(dec_trees_pred, factor(validation$classe))
trees_pred
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1519  473  484  451  156
##          B   28  355   45   10  130
##          C   83  117  423  131  131
##          D   40  194   74  372  176
##          E    4    0    0    0  489
##
## Overall Statistics
##
##                Accuracy : 0.5366
##                  95% CI : (0.5238, 0.5494)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3957
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9074  0.31168  0.41228  0.38589  0.45194
## Specificity            0.6286  0.95512  0.90492  0.90165  0.99917
## Pos Pred Value         0.4927  0.62500  0.47797  0.43458  0.99189
## Neg Pred Value         0.9447  0.85255  0.87940  0.88228  0.89002
## Prevalence             0.2845  0.19354  0.17434  0.16381  0.18386
## Detection Rate         0.2581  0.06032  0.07188  0.06321  0.08309
## Detection Prevalence   0.5239  0.09652  0.15038  0.14545  0.08377
## Balanced Accuracy      0.7680  0.63340  0.65860  0.64377  0.72555
```

The accuracy, 95th percentile confidence interval, P-value and other parameters are shown. Other features such as the sensitivity (true positive rate), the specificity, positive predicted value for each of the classes can be obtained. It is observed that the model most accurately predicts class A (highest sensitivity). It means that the model is correctly able to identify the true positives from the data belonging to class A. However, the ability of the model to correctly identify negative cases is low for class A compared to the other classes.

SVM

Support Vector Machine (SVM) is a powerful and popular machine learning algorithm primarily used for classification, but it can also be applied to regression and outlier detection. The core idea behind SVM is to find the best boundary, called a hyperplane, that separates different classes of data points in such a way that the margin between the classes is maximized. Support Vector Machines are highly flexible and powerful machine learning algorithms that excel at classification tasks and can handle complex, non-linear relationships through kernel methods. Although they require careful tuning and can be computationally expensive, they are widely used due to their ability to create well-defined decision boundaries and generalize well to unseen data.

```
SVM <- train(classe~., data=training, method="svmLinear", trControl = control, tuneLength = 5, verbose =
SVM_Predictions <- predict(SVM, validation)
Pred_SVN <- confusionMatrix(SVM_Predictions, factor(validation$classe))
Pred_SVN
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1537  154   79   69   50
##          B   29  806   90   46  152
##          C   40   81  797  114   69
##          D   61   22   32  697   50
##          E    7   76   28   38  761
##
## Overall Statistics
##
##                Accuracy : 0.7813
##                  95% CI : (0.7705, 0.7918)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.722
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9182   0.7076   0.7768   0.7230   0.7033
## Specificity            0.9164   0.9332   0.9374   0.9665   0.9690
## Pos Pred Value         0.8137   0.7177   0.7239   0.8086   0.8363
## Neg Pred Value         0.9657   0.9301   0.9521   0.9468   0.9355
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2612   0.1370   0.1354   0.1184   0.1293
## Detection Prevalence   0.3210   0.1908   0.1871   0.1465   0.1546
## Balanced Accuracy      0.9173   0.8204   0.8571   0.8447   0.8362
```

Compared to Decision trees, it is observed that support vector machining provides superior statistic parameters across the prediction of almost all the classes and is subsequently chosen as the prediction algorithm to predict the classes in the test data.

Test Data Predictions

```
Test_Predictions <- predict(SVM, testingData)
print(Test_Predictions)
```

```
##  [1] C A B C A E D D A A C A B A E E A B B B
## Levels: A B C D E
```

It is seen to correctly predict the classe (5 levels) outcome for 20 casses with the SVM model.