# Prediction assignment

## Installing packages:

```
library(lattice)
library(ggplot2)
library(caret)
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rpart)
library(rpart.plot)
```

## Set seed:

```
set.seed(1000)
```

## Load data and deal with “NA”,“#DIV/0!”and “”.

```
url.train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url.test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(url.train), na.strings = c("NA", "", "#DIV0!"))
testing <- read.csv(url(url.test), na.strings = c("NA", "", "#DIV0!"))
```

## Exploratory data analysis to the training -data set.

```
dim(training)
```

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

```
colnames(training)
```

```
##   [1] "X"                         "user_name"
##   [3] "raw_timestamp_part_1"      "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"            "new_window"
##   [7] "num_window"                "roll_belt"
##   [9] "pitch_belt"                "yaw_belt"
##  [11] "total_accel_belt"          "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"       "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"        "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"         "max_roll_belt"
##  [19] "max_picth_belt"            "max_yaw_belt"
##  [21] "min_roll_belt"             "min_pitch_belt"
##  [23] "min_yaw_belt"              "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"      "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"      "avg_roll_belt"
##  [29] "stddev_roll_belt"          "var_roll_belt"
##  [31] "avg_pitch_belt"            "stddev_pitch_belt"
##  [33] "var_pitch_belt"            "avg_yaw_belt"
##  [35] "stddev_yaw_belt"           "var_yaw_belt"
##  [37] "gyros_belt_x"              "gyros_belt_y"
##  [39] "gyros_belt_z"              "accel_belt_x"
##  [41] "accel_belt_y"              "accel_belt_z"
##  [43] "magnet_belt_x"             "magnet_belt_y"
##  [45] "magnet_belt_z"             "roll_arm"
##  [47] "pitch_arm"                 "yaw_arm"
##  [49] "total_accel_arm"           "var_accel_arm"
##  [51] "avg_roll_arm"              "stddev_roll_arm"
##  [53] "var_roll_arm"              "avg_pitch_arm"
##  [55] "stddev_pitch_arm"          "var_pitch_arm"
##  [57] "avg_yaw_arm"               "stddev_yaw_arm"
##  [59] "var_yaw_arm"               "gyros_arm_x"
##  [61] "gyros_arm_y"               "gyros_arm_z"
##  [63] "accel_arm_x"               "accel_arm_y"
##  [65] "accel_arm_z"               "magnet_arm_x"
##  [67] "magnet_arm_y"              "magnet_arm_z"
##  [69] "kurtosis_roll_arm"         "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"          "skewness_roll_arm"
##  [73] "skewness_pitch_arm"        "skewness_yaw_arm"
##  [75] "max_roll_arm"              "max_picth_arm"
##  [77] "max_yaw_arm"               "min_roll_arm"
##  [79] "min_pitch_arm"             "min_yaw_arm"
##  [81] "amplitude_roll_arm"        "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"         "roll_dumbbell"
##  [85] "pitch_dumbbell"            "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"    "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"     "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell"   "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"         "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"          "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"        "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"   "amplitude_pitch_dumbbell"
## [101] "amplitude_yaw_dumbbell"    "total_accel_dumbbell"
## [103] "var_accel_dumbbell"        "avg_roll_dumbbell"
```

```
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

```
str(training[1:7])
```

```
## 'data.frame':    19622 obs. of  7 variables:
##  $ X                   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name           : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2
2 2 2 2 ...
##  $ raw_timestamp_part_1: int  1323084231 1323084231 1323084231 1323084232 132308
4232 1323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2: int  788290 808298 820366 120339 196328 304277 368296 4
40390 484323 484434 ...
##  $ cvtd_timestamp      : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9
9 9 9 9 ...
##  $ new_window          : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window          : int  11 11 11 12 12 12 12 12 12 12 ...
```

So there are 160 variables. The first 7 variables can be left out. And all there columns where all the values are missing, can be left out. Let’s do that and see, which variables are left.

```
training   <-training[,-c(1:7)]
testing <-testing[,-c(1:7)]
training<-training[,colSums(is.na(training)) == 0]
testing <-testing[,colSums(is.na(testing)) == 0]
colnames(training)
```

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

# Create data partition

75% of the data goes to the training set and 25% to the testing set.

```
traintrainset <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
TrainTraining <- training[traintrainset, ]
TestTraining <- training[-traintrainset, ]
```

Lets look closer at the classe -variable in the TrainTraining data set:

```
str(TrainTraining$classe)
```

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

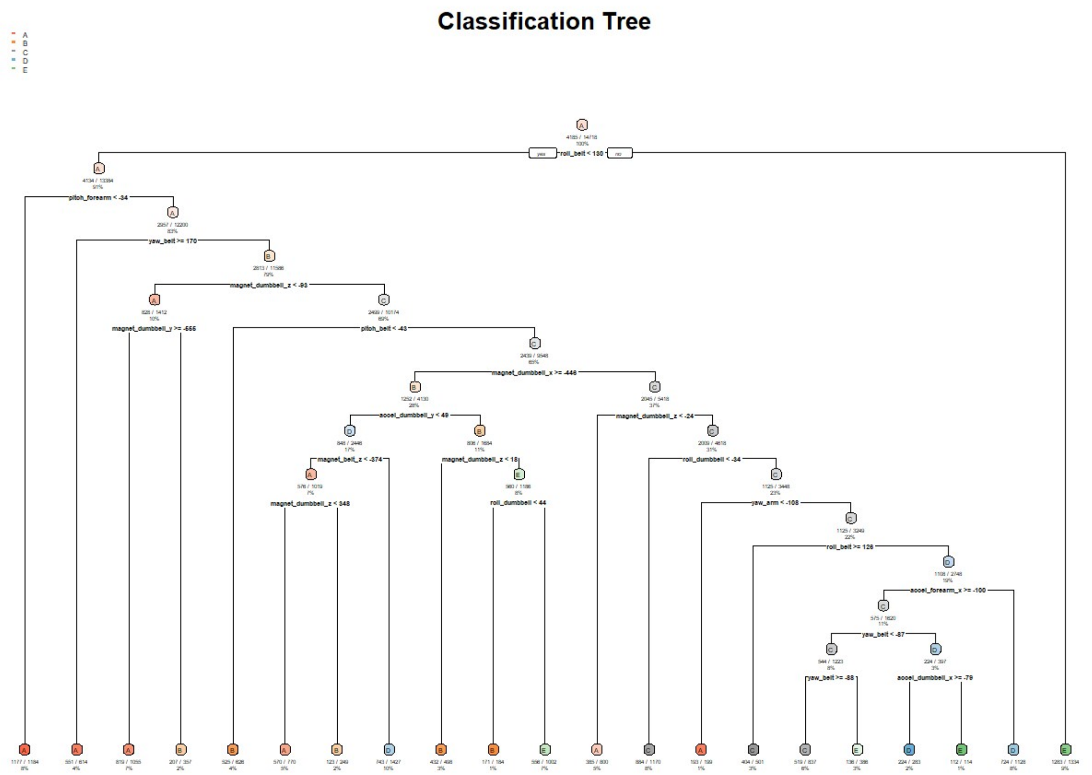
```
summary(TrainTraining$classe)
```

```
##    A    B    C    D    E
## 4185 2848 2567 2412 2706
```

So classe is a factor variable with 5 levels.

# Decision tree

```
tree1 <- rpart(classe ~ ., data=TrainTraining, method="class")
rpart.plot(tree1, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```




Test results on the testing data.

```
prediction1 <- predict(tree1, TestTraining, type = "class")
confusionMatrix(prediction1, TestTraining$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1207  154   40   54   61
##          B   40  499   35   61   23
##          C   60   98  563   52   43
##          D   46  104  188  554   97
##          E   42   94   29   83  677
## 
## Overall Statistics
## 
##                Accuracy : 0.7137
##                  95% CI : (0.7008, 0.7263)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.6373
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8652   0.5258   0.6585   0.6891   0.7514
## Specificity            0.9119   0.9598   0.9375   0.8939   0.9380
## Pos Pred Value         0.7962   0.7584   0.6900   0.5602   0.7319
## Neg Pred Value         0.9445   0.8940   0.9286   0.9361   0.9437
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2461   0.1018   0.1148   0.1130   0.1381
## Detection Prevalence   0.3091   0.1342   0.1664   0.2017   0.1886
## Balanced Accuracy      0.8886   0.7428   0.7980   0.7915   0.8447
```

The accuracy of a decision tree is 0.7137.

# Random forest

```
forest2 <- randomForest(classe ~. , data=TrainTraining, method="class")
```

Let’s test this on training set.

```
prediction2 <- predict(forest2, TestTraining, type = "class")
confusionMatrix(prediction2, TestTraining$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    1    0    0    0
##          B    2  947    7    0    0
##          C    0    1  847    5    0
##          D    0    0    1  796    3
##          E    0    0    0    3  898
##
## Overall Statistics
##
##                Accuracy : 0.9953
##                  95% CI : (0.993, 0.997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9941
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9979   0.9906   0.9900   0.9967
## Specificity            0.9997   0.9977   0.9985   0.9990   0.9993
## Pos Pred Value         0.9993   0.9906   0.9930   0.9950   0.9967
## Neg Pred Value         0.9994   0.9995   0.9980   0.9981   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1931   0.1727   0.1623   0.1831
## Detection Prevalence   0.2843   0.1949   0.1739   0.1631   0.1837
## Balanced Accuracy      0.9991   0.9978   0.9946   0.9945   0.9980
```

The accuracy of a random forest is 0.9953.

# Conclusion:

The accuracy of a random forest (0.9953) is better than of a decision tree (0.7137), and that’s why random forest is chosen to count the final results.

# Final results:

Let’s count the final results with the random forest -model.

```
prediction <- predict(forest2, testing, type="class")
prediction
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```