Practical Machine Learning Course Project

Corey L

8/1/2019

```r
library(caret)
library(visdat)
library(gbm)
library(dplyr)
library(parallel)
library(doParallel)
library(beepr)
set.seed(42) # The answer to everything.
```

Getting the data,

Reading the comma separated values into objects

```r
#Creating a data folder in the working dirctory and downloading the datasets for the assignment
 if(!file.exists("./data")){dir.create("./data")}
        fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
        download.file(fileUrl,destfile="./data/training.csv")
        fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
        download.file(fileUrl,destfile="./data/testing.csv")
```

```r
datatrain <- read.csv("./data/training.csv", na.strings=c('NA','','#DIV/0!')) # Training Data
datatest <- read.csv("./data/testing.csv", na.strings=c('NA','','#DIV/0!'))   # Test Data
```

Exploratory Analysis

Dimensions of the dataset

```r
dim(datatrain)
```

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

The training dataset has 19622 obs. of 160 variables

The target variable for prediction in this database are listed in the `classe` column

```r
str(datatrain$classe)
```

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

Unique values in datatrain$classe

```r
unique(datatrain$classe)
```

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

List of all column names in the dataset

```r
colnames(datatrain)
```

```
##   [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"
```

Data Cleaning

The first seven columns of the datasets contain user names, timecodes and data that is irrelevant to predicting exercises. for prediction model training, the columns will be removed. Also within the datasets are `NA` and non-zero values. The invalid values are not compatible with prediction models and will be removed by eliminating the columns that contain them as a majority. Each cleaning process applied to the training dataset will also be applied to the testing dataset.

Observing the NA, and invalid data in the training dataset after removing unnecessary columns

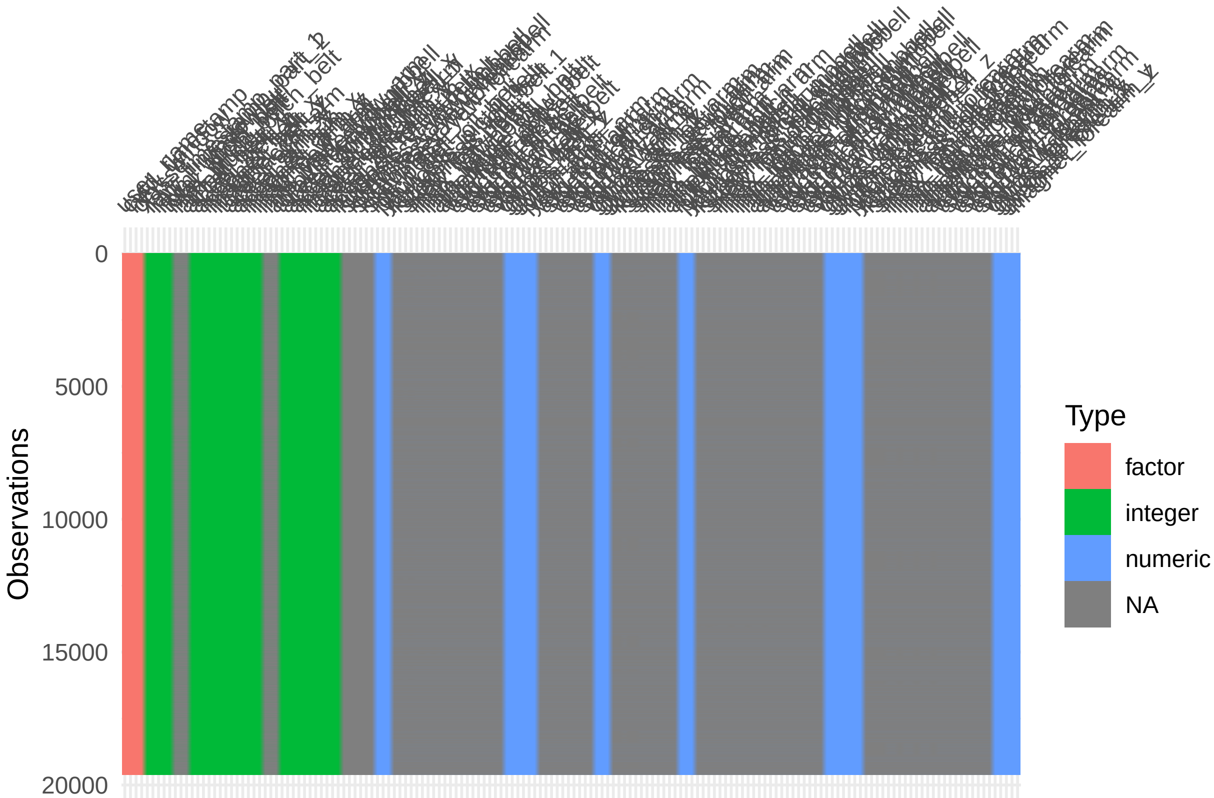
```r
table(is.na(datatrain))
```

```
##
##   FALSE    TRUE
## 1214418 1925102
```

There are `1,925,102` NA values in the dataset.

Visualizing the amount of missing data in the training dataset. Column names are overlaying due to the number of the columns, but the graph performs a function of representing missing data.

```r
vis_dat(datatrain, warn_large_data = FALSE)
```



From observing the graph above, we can see that there is very little valid data in the areas populated with many `na` values. The loss of this data will have a minimal impact on training our models. The columns that possess more than `90%` of their values as `NA` will be removed.

```r
colrmv <- which(colSums(is.na(datatrain) |datatrain=="")>0.9*dim(datatrain)[1])
datatraincl <- datatrain[,-colrmv]
datatraincl <- datatraincl[,-c(1:7)]
```

Dimensions of the new dataset with columns removed

```r
dim(datatraincl)
```

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

Columns to be be considered for the prediction model for having invalid, or NA values

```r
str(datatraincl)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ 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 ...
##  $ 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 ...
##  $ 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 ...
##  $ 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 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

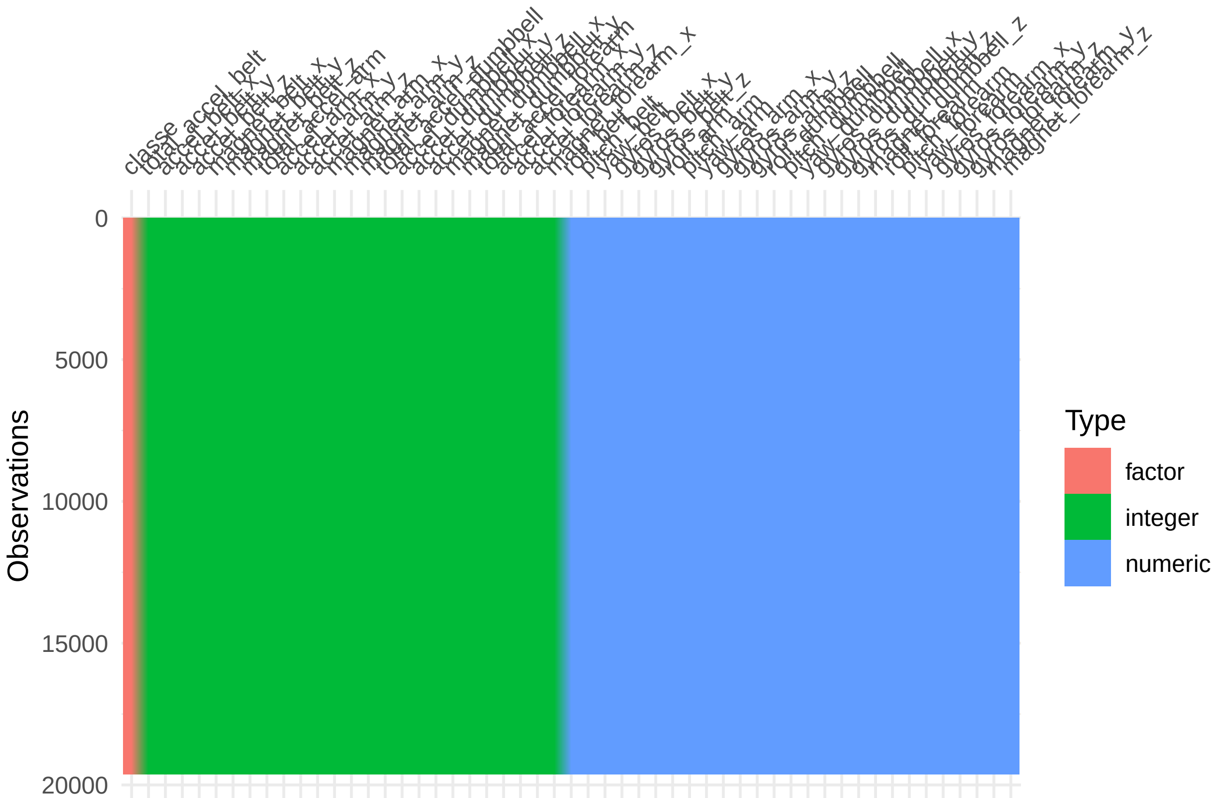
Observing NA in the cleaned dataset

```r
table(is.na(datatraincl))
```

```
##
##   FALSE
## 1039966
```

Visualizing the cleaned training dataset

```r
vis_dat(datatraincl, warn_large_data = FALSE)
```



Cross validation of within the training dataset

Splitting the `datatrain` set into testing and training variables, 70% training split.

```
inTrain = createDataPartition(y=datatraincl$classe, p = 0.7, list=FALSE)
training = datatraincl[inTrain,]
testing = datatraincl[-inTrain,]
```

Setting cross validation parameters for the following models

```
fitControl <- trainControl(method='cv',
                number = 3,
                allowParallel = TRUE)
```

Training Prediction Models

Training a Random Forest Model

```
rfmod <- train(classe ~.,
               method="rf",
               data=training,
               trControl=fitControl)
rfmod
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9157, 9157, 9160
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9871155  0.9836985
##   27    0.9891530  0.9862771
##   52    0.9827466  0.9781705
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Training a Stochastic Gradient Boosted Model

```
gbmmod <- train(classe ~.,
              data = training,
              method = "gbm",
              trControl = fitControl,
              verbose = FALSE)
gbmmod
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9158, 9159, 9157
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7563524  0.6910827
##   1                  100      0.8164088  0.7676320
##   1                  150      0.8517869  0.8124402
##   2                   50      0.8549170  0.8161499
##   2                  100      0.9044906  0.8790751
##   2                  150      0.9288778  0.9099858
##   3                   50      0.8956100  0.8678187
##   3                  100      0.9401613  0.9242632
##   3                  150      0.9584331  0.9474127
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Training a Bagged CART Model

```
bcmod <- train(classe ~.,
              data = training,
              method = "treebag",
```

```
              trControl = fitControl)
bcmod
```

```
## Bagged CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9158, 9158, 9158
## Resampling results:
##
##   Accuracy   Kappa
##   0.9807818  0.9756915
```

Training with Quinlan's C5.0 algorithm, which uses both basic-tree, and rules based models

```
c50mod <- train(classe ~.,
              data = training,
              method = "C5.0",
              trControl = fitControl)
c50mod
```

```
## C5.0
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9157, 9159, 9158
## Resampling results across tuning parameters:
##
##   model  winnow  trials  Accuracy   Kappa
##   rules  FALSE    1      0.9489695  0.9354649
##   rules  FALSE   10      0.9870422  0.9836071
##   rules  FALSE   20      0.9910459  0.9886726
##   rules   TRUE    1      0.9491877  0.9357233
##   rules   TRUE   10      0.9882068  0.9850808
##   rules   TRUE   20      0.9911916  0.9888568
##   tree   FALSE    1      0.9433638  0.9283597
##   tree   FALSE   10      0.9852948  0.9813970
##   tree   FALSE   20      0.9906093  0.9881205
##   tree    TRUE    1      0.9435095  0.9285394
##   tree    TRUE   10      0.9868237  0.9833327
##   tree    TRUE   20      0.9895900  0.9868309
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were trials = 20, model = rules
##  and winnow = TRUE.
```

Cross Validation with the testing dataset

Comparing the the three model's performance against the `testing` dataset. The accuracy of each will be presented as a table. For each model, the `testing` dataset was used to assess prediction performance, and its output was stored as an object. The confusion matrix of the resultant object was used to predict the `$classe` variable in the dataset. Accuracy of all model predictions were stored as a set and presented for observation.

```
predRF <- predict(rfmod, newdata=testing)
cmRF <- confusionMatrix(predRF, testing$classe)
predGBM <- predict(gbmmod, newdata=testing)
cmGBM <- confusionMatrix(predGBM, testing$classe)
predc50 <- predict(c50mod, newdata=testing)
cmc50 <- confusionMatrix(predc50, testing$classe)
AccuracyResults <- data.frame(
  Model = c('RF', 'GBM', 'C5.0'),
  Accuracy = rbind(cmRF$overall[1], cmGBM$overall[1], cmc50$overall[1])
)
print(AccuracyResults)
```

```
##   Model  Accuracy
## 1    RF 0.9949023
## 2   GBM 0.9634664
## 3  C5.0 0.9960918
```

From the table above, we can see the Quinlan's C5.0 algorithm `C50` has the highest accuracy rate of `99.6%`. For cross validation of the training sample, we will use this model on the provided testing set.

Results and Proof of Prediction

Using the best model `c50mod` on the training set.

```
predictTEST <- predict(c50mod, newdata=datatest)
predictTEST
```

```
##  [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
```

From this output, the final course quiz was completed with a graded score of `100%`.

```
beep("coin") # Signals file processing is complete, X3 for effect
beep("coin")
beep("coin")
```

Appendix

Data Source

Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6. Cited by 2 (Google Scholar)