Prediction Assignment

HONAKAYU

24/02/2021

Load libraries

```
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(corrplot)
library(gbm)
library(parallel)
library(doParallel)
library(e1071)
library(visdat)
library(parallel)
```

Load the training & test data

```
train_source <- read.csv('http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv', header='
test_source <- read.csv('http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv', header=TRU
dim(train_source)
```

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

```
dim(test_source)
```

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

There are 19622 observations in the training data and 160 variables. The test data only contains 20 observations and also 160 variables.

Can we remove any variables in order to simplify model?

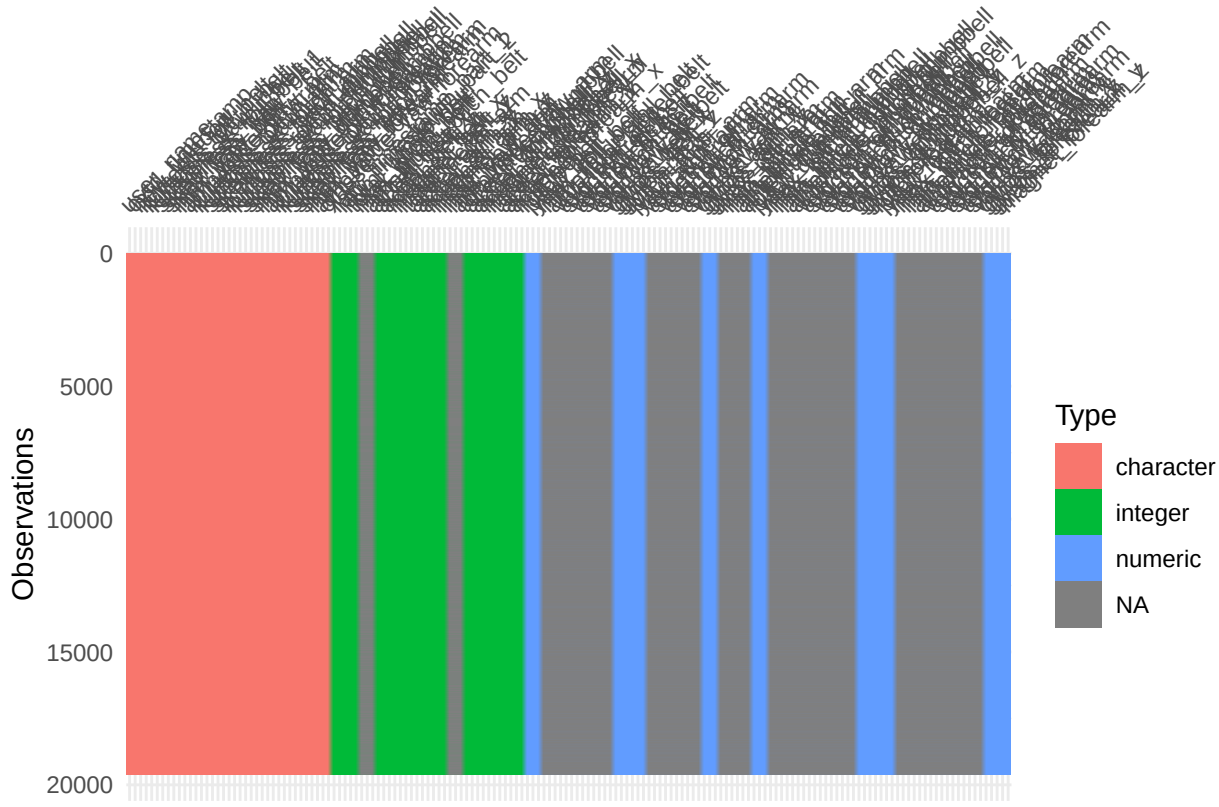
Actual specific usage data line the given files only start at variable number 8. The ones prior will intuitively not help us predict how well they do the exercise (classe) so variables 1 to 7 will be removed.

```
table(is.na(train_source))
```

```
##
##   FALSE    TRUE
## 1852048 1287472
```

There's a significant amount of NA values in the data (based on our data load criteria). 1,925,192 to be exact. These will need to be removed. Visualisation below.

```
vis_dat(train_source, warn_large_data = FALSE, palette="default")
```



Let's remove any variables which have more than 95% of their data as NA and as discussed before, also remove the first 7 variables.

```
# Remove variables with 'Nearly Zero Variance'
nzv <- nearZeroVar(train_source)
train_source  <- train_source[, -nzv]
test_source  <- test_source[, -nzv]

# Remove NA values
mostlyNA <- sapply(train_source, function(x) mean(is.na(x))) > 0.95
train_source <- train_source[, mostlyNA == FALSE]
test_source <- test_source[, mostlyNA == FALSE]

# Remove first 7 variables
```

```
train_source <- train_source[, -c(1:7)]
test_source <- test_source[, -c(1:7)]
```

This will leave us with 52 variables.

Splitting the data for modeling.

We partition the training dataset (train_Data_source) into two sets: 60% of the training data for the modeling process and the remaining 40% for the test set. The other data file (test_source) is not modified and will be used to predict the right answers for the quiz.

```
in_train  <- createDataPartition(train_source$classe, p = 0.6, list = FALSE)
train_set <- train_source[in_train, ]
test_set  <- train_source[-in_train, ]
```

Building the prediction model

We will build 3 models. 1) Decision Tree 2) Random Forest 3) Generalised Boosted Model. Each analysis will contain a confusion matrix which will help visualise the predictions. We will pick the model with the highest accuracy to predict the correct answers for the quiz.

Cross validation

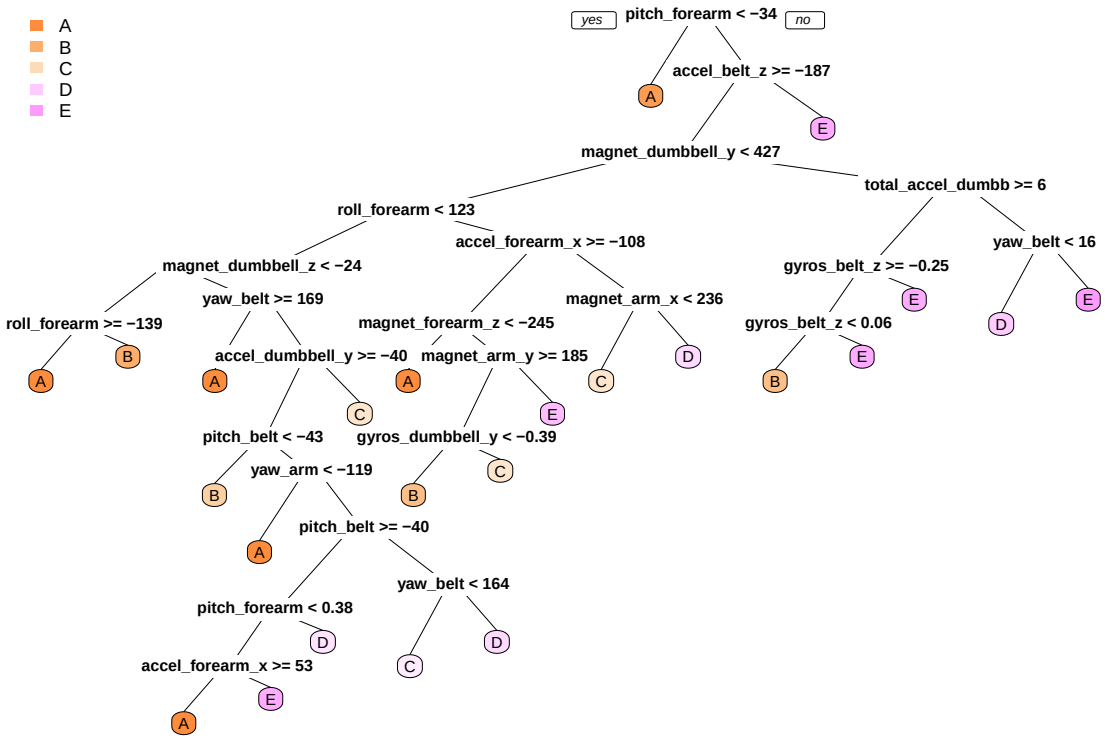
Here we set the cross-validation parameters for the models:

```
# decision tree
fitControlDT <- rpart.control(method = "cv", number = 3, verboseIter = FALSE)

# random forest
fitControlRF <- trainControl(method = "cv", number = 3, verboseIter = FALSE, allowParallel = TRUE)

# generalized boosted model
fitControlGBM <- trainControl(method = "repeatedcv", number = 3, repeats = 1, verboseIter = FALSE, allo
```

Decision Tree

Build the model and the corresponding decision tree diagram.

```
set.seed(1234)
mod_fit_dt <- rpart(as.factor(classe) ~ .,data = train_set,control = fitControlDT, method = "class")

prp(mod_fit_dt, faclen = 0, box.palette = "OrPu", cex = 0.50, legend.x = 0, legend.y = 1, legend.cex =
```



Validate the Decision Tree model on the test set (testSet) to determine how well it performed and the accuracy of the results.

```
prediction_dt <- predict(mod_fit_dt, newdata = test_set, type = "class")
conf_matrix_dt <- confusionMatrix(factor(prediction_dt), factor(test_set$classe))
print(conf_matrix_dt)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1957  227   25   48   33
##          B   78  814   90  106  148
##          C   47  160  814   90   80
##          D  126  219  202  916  199
##          E   24   98  237  126  982
##
## Overall Statistics
##
##                Accuracy : 0.6988
##                  95% CI : (0.6885, 0.709)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6192
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8768   0.5362   0.5950   0.7123   0.6810
## Specificity            0.9407   0.9333   0.9418   0.8863   0.9243
## Pos Pred Value         0.8546   0.6586   0.6835   0.5511   0.6694
## Neg Pred Value         0.9505   0.8935   0.9168   0.9402   0.9279
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2494   0.1037   0.1037   0.1167   0.1252
## Detection Prevalence   0.2919   0.1575   0.1518   0.2118   0.1870
## Balanced Accuracy      0.9087   0.7348   0.7684   0.7993   0.8026
```

The results from the confusion matrix for the Decision Tree model show a predicted accuracy of 0.7 giving an out-of-sample error rate of 0.3 which is high.

Random Forest

Here we build the random forest model and we hope to see better results.

```
set.seed(1234)

mod_fit_rf <- train(classe ~ .,method="rf", data=train_source, trControl=fitControlRF, verbose = FALSE)

print(mod_fit_rf$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, verbose = FALSE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 26
##
##         OOB estimate of  error rate: 0.41%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5576    3    0    0    1 0.0007168459
## B   20 3770    6    0    1 0.0071108770
## C    0   11 3404    7    0 0.0052600818
## D    0    0   22 3192    2 0.0074626866
## E    0    0    5    3 3599 0.0022179096
```

```
PredictionRF <- predict(mod_fit_rf, newdata = test_set)
conf_matrix_rf <- confusionMatrix(factor(PredictionRF), factor(test_set$classe))
print(conf_matrix_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 2232    0    0    0    0
##          B    0 1518    0    0    0
##          C    0    0 1368    0    0
##          D    0    0    0 1286    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9995, 1)
##     No Information Rate : 0.2845
##     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             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

These are excellent prediction results with an accuracy of 1 and a confidence interval of 99.9%. There is a neglible out of sample error.

No other models wil be explored due to the excellent results derived from the random forest model.

```
predictionQuiz <- predict(mod_fit_rf, newdata = test_source)
predictionQuiz
```

```
##  [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
```

These are our quiz predictions which resulted in a score of 100%.