Machine Learning Project

Summary

In this project we will take a data related to the quality of weight lifting performed by a group of people and predict the quality of weight lifting of another set of data.

Data preparation

We first will load the data into a few tables then trim out the noise

```
require(data.table)
```

```
## Loading required package: data.table
```

```
require(randomForest)
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1337)

# load libraries
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
# "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
training_url <- "pml-training.csv"
# "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
testing_url <- "pml-testing.csv"
training <- read.csv(training_url, na.strings = c("NA", ""), strip.white=T)
testing <- read.csv(testing_url, na.strings = c("NA", ""), strip.white=T)
set_names <- names(training)
# summary(training)
# some worthless columns
worthless_column_indexes <- grep("X|user_name|.*timestamp.*", set_names)
training <- training[, -worthless_column_indexes]

# remove na item columns
na_items_index <- apply(training, 2, function(x) { sum(is.na(x)) })
training <- training[, which(na_items_index == 0)]

# remove columns that have low variability in order to train better
near_zero_variability_column_indexes <- nearZeroVar(training)
training <- training[, -near_zero_variability_column_indexes]
```

Data partitioning

We split the data into two parts, a training and testing part. For this part I did an 80/20 split between training and testing

```
train_idx <- createDataPartition(training$classe, p=0.8, list=F)
t_training <- training[train_idx,]
t_testing <- training[-train_idx,]
```

Training

Now we'll create a random forrest training model for use on our partitioned test set.

```
# for performance, just run this the first time
if (!exists("train_model")){
  train_control <- trainControl(allowParallel=T, method="cv", number=4)
  train_model <- train(classe ~ ., data=t_training, model="rf", trControl=train_control)
}

train_model
```

```
## Random Forest
##
## 15699 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 11774, 11775, 11773, 11775
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9939488  0.9923452
##   27    0.9969425  0.9961325
##   53    0.9950952  0.9937957
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
train_model$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, model = "rf")
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.19%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4462    1    0    0    1 0.0004480287
## B    5 3030    2    1    0 0.0026333114
## C    0    7 2731    0    0 0.0025566107
## D    0    0    6 2566    1 0.0027205597
## E    0    0    0    6 2880 0.0020790021
```

```
round(max(train_model$results$Accuracy), 4) * 100
```

```
## [1] 99.69
```

Our Training model has an accuracy of 99.73%.

Testing our model

Let’s test our training model against our partitioned test set

```
# test out our training model on the testing portion
predict_model <- predict(train_model, newdata=t_testing)

# let's check out the confusion matrix
confusion_matrix <- confusionMatrix(t_testing$classe, predict_model)
confusion_matrix$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    0  758    1    0    0
##          C    0    0  684    0    0
##          D    0    0    2  641    0
##          E    0    0    0    3  718
```

```
# calculate our performance across the resample
postResample(predict_model, t_testing$classe)
```

```
##  Accuracy     Kappa
## 0.9984706 0.9980655
```

In our testing model we got an accuracy of 99.8% which was extremely accurate

Applying our model for the quiz

```
predict_quiz <- predict(train_model, newdata=testing)

predict_quiz
```

```
##  [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
```