Practical Machine Learning Prediction Project

Balaji

Prepare the datasets

Load the training data into a data table.

```
echo = TRUE
library(data.table)
library(utils)
url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
D <- fread(url)
```

Load the testing data into a data table.

```
url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
TestSet <- fread(url)
```

Which variables in the test dataset have zero `NA`s?

Belt, arm, dumbbell, and forearm variables that do not have any missing values in the test dataset will be **predictor candidates**.

```
isAnyMissing <- sapply(TestSet, function (x) any(is.na(x) | x == ""))
isPredictor <- !isAnyMissing & grepl("belt|[^(fore)]arm|dumbbell|forearm", names(isAnyMissing))
predCandidates <- names(isAnyMissing)[isPredictor]
predCandidates
```

```
##  [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"
```

Subset the primary dataset to include only the **predictor candidates** and the outcome variable, `classe`.

```
varToInclude <- c("classe", predCandidates)
D <- D[, varToInclude, with=FALSE]
dim(D)
```

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

```
names(D)
```

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

Make `classe` into a factor.

```
D <- D[, classe := factor(D[, classe])]
D[, .N, classe]
```

```
##    classe    N
## 1:      A 5580
## 2:      B 3797
## 3:      C 3422
## 4:      D 3216
## 5:      E 3607
```

Split the dataset into a 60% training and 40% probing dataset.

```
library(caret)
```

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

```
seed <- as.numeric(as.Date("2014-10-26"))
set.seed(seed)
inTrain <- createDataPartition(D$classe, p=0.6)
DTrain <- D[inTrain[[1]]]
DCV <- D[-inTrain[[1]]]
```

Preprocess the prediction variables by centering and scaling.

```
X <- DTrain[, predCandidates, with=FALSE]
preProc <- preProcess(X)
preProc
```

```
##
## Call:
## preProcess.default(x = X)
##
## Created from 11776 samples and 52 variables
## Pre-processing: centered, scaled
```

```
XCS <- predict(preProc, X)
DTrainCS <- data.table(data.frame(classe = DTrain[, classe], XCS))
```

Apply the centering and scaling to the probing dataset.

```
X <- DCV[, predCandidates, with=FALSE]
XCS <- predict(preProc, X)
DCVCS <- data.table(data.frame(classe = DCV[, classe], XCS))
```

Check for near zero variance.

```
nzv <- nearZeroVar(DTrainCS, saveMetrics=TRUE)
if (any(nzv$nzv)) nzv else message("No variables with near zero variance")
```

```
## No variables with near zero variance
```

Train a prediction model

I chose to use random forests for a prediction model The error will be estimated using the 40% probing sample.

Fit model over the tuning parameters.

```
#system.time(trainingModel <- train(classe ~ ., data=DTrainCS, method="rf"))
if (file.exists("trainingModel.RData")) {
  load("trainingModel.RData")
  } else
    trainingModel <- train(classe ~ ., data=DTrainCS, method="rf")
```

Evaluate the model on the training dataset

```
trainingModel
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9859564  0.9822356  0.002191021  0.002775850
##   27    0.9871433  0.9837386  0.001722883  0.002178948
##   52    0.9772794  0.9712595  0.004389185  0.005565894
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
phat <- predict(trainingModel, DTrainCS)
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
confusionMatrix(phat, DTrain[, classe])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     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             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

The training model seems to perform with 100% accuracy. This could either be a really good model, or we might have over-fit the data. Let us explore using a cross-validation dataset.

Evaluate the model on the cross-validation dataset

```
phat <- predict(trainingModel, DCVCS)
confusionMatrix(phat, DCVCS[, classe])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230   16    0    0    0
##          B    1 1498    7    2    2
##          C    0    4 1351   17    7
##          D    0    0   10 1266    7
##          E    1    0    0    1 1426
##
## Overall Statistics
##
##                Accuracy : 0.9904
##                  95% CI : (0.988, 0.9925)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9879
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9868   0.9876   0.9844   0.9889
## Specificity            0.9971   0.9981   0.9957   0.9974   0.9997
## Pos Pred Value         0.9929   0.9921   0.9797   0.9867   0.9986
## Neg Pred Value         0.9996   0.9968   0.9974   0.9970   0.9975
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1909   0.1722   0.1614   0.1817
## Detection Prevalence   0.2863   0.1925   0.1758   0.1635   0.1820
## Balanced Accuracy      0.9981   0.9925   0.9916   0.9909   0.9943
```

Good News! The out-of-sample error should hopefully be less than 1%

Display the final model

```
varImp(trainingModel)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt            100.000
## pitch_forearm         61.969
## yaw_belt              55.124
## pitch_belt            47.587
## roll_forearm          44.764
## magnet_dumbbell_y     44.730
## magnet_dumbbell_z     43.871
## accel_dumbbell_y      20.989
## roll_dumbbell         19.308
## magnet_dumbbell_x     18.501
## accel_forearm_x       17.435
## accel_belt_z          16.504
## magnet_belt_z         14.958
## total_accel_dumbbell  14.832
## magnet_forearm_z      14.439
## accel_dumbbell_z      13.625
## magnet_belt_y         13.204
## yaw_arm               12.038
## gyros_belt_z          11.938
## magnet_belt_x          9.393
```

```
trainingModel$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.77%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3342    5    1    0    0 0.001792115
## B   16 2254    7    1    1 0.010969724
## C    0   10 2037    7    0 0.008276534
## D    1    1   27 1898    3 0.016580311
## E    0    2    2    7 2154 0.005080831
```

We see that the estimated error rate is less than 1%

Save training model object for later.

```
save(trainingModel, file="trainingModel.RData")
```

Predict on the test data

Load the training model.

```
load(file="trainingModel.RData", verbose=TRUE)
```

```
## Loading objects:
##   trainingModel
```

Get predictions and evaluate.

```
TestSetCS <- predict(preProc, TestSet[, predCandidates, with=FALSE])
hat <- predict(trainingModel, TestSetCS)
TestSet <- cbind(hat , TestSet)
subset(TestSet, select=names(TestSet)[grep("belt|[^(fore)]arm|dumbbell|forearm", names(TestSet), invert=
```

```
##     hat V1 user_name raw_timestamp_part_1 raw_timestamp_part_2
##  1:   B  1     pedro           1323095002               868349
##  2:   A  2    jeremy           1322673067               778725
##  3:   B  3    jeremy           1322673075               342967
##  4:   A  4    adelmo           1322832789               560311
##  5:   A  5    eurico           1322489635               814776
##  6:   E  6    jeremy           1322673149               510661
##  7:   D  7    jeremy           1322673128               766645
##  8:   B  8    jeremy           1322673076                54671
##  9:   A  9  carlitos           1323084240               916313
## 10:   A 10   charles           1322837822               384285
## 11:   B 11  carlitos           1323084277                36553
## 12:   C 12    jeremy           1322673101               442731
## 13:   B 13    eurico           1322489661               298656
## 14:   A 14    jeremy           1322673043               178652
## 15:   E 15    jeremy           1322673156               550750
## 16:   E 16    eurico           1322489713               706637
## 17:   A 17     pedro           1323094971               920315
## 18:   B 18  carlitos           1323084285               176314
## 19:   B 19     pedro           1323094999               828379
## 20:   B 20    eurico           1322489658               106658
##       cvtd_timestamp new_window num_window problem_id
##  1: 05/12/2011 14:23         no         74          1
##  2: 30/11/2011 17:11         no        431          2
##  3: 30/11/2011 17:11         no        439          3
##  4: 02/12/2011 13:33         no        194          4
##  5: 28/11/2011 14:13         no        235          5
##  6: 30/11/2011 17:12         no        504          6
##  7: 30/11/2011 17:12         no        485          7
##  8: 30/11/2011 17:11         no        440          8
##  9: 05/12/2011 11:24         no        323          9
## 10: 02/12/2011 14:57         no        664         10
```

```
## 11: 05/12/2011 11:24         no        859         11
## 12: 30/11/2011 17:11         no        461         12
## 13: 28/11/2011 14:14         no        257         13
## 14: 30/11/2011 17:10         no        408         14
## 15: 30/11/2011 17:12         no        779         15
## 16: 28/11/2011 14:15         no        302         16
## 17: 05/12/2011 14:22         no         48         17
## 18: 05/12/2011 11:24         no        361         18
## 19: 05/12/2011 14:23         no         72         19
## 20: 28/11/2011 14:14         no        255         20
```

Submission to Coursera

Write submission files to `PMLfiles/`.

```
save_files = function(x){
  n = length(x)
  path <- "PMLfiles/"
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=file.path(path, filename),quote=FALSE,row.names=FALSE,col.names=FALSE)
    }
  }
save_files(hat)
```