HAR_Analytics

Rama Tripathy

May 20, 2017

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

In this project, the goal is to to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and predict the manner in which they did the exercise. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset). The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Data Preprocessing

1. Load libraries and setup root directory
2. Load Data

```
# Data sources
trainurl <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'     # Train Data
testurl <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'       # Test Data
download.file(trainurl,destfile= 'train.csv', method="curl")
download.file(testurl, destfile = 'test.csv', method="curl")
train <- read.csv("train.csv")
test <- read.csv("test.csv")
dim(train)
```

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

```
dim(test)
```

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

3. Select columns for belt, forearm, arm, and dumbell

```
target <- train$classe
train <- select(train,  contains("arm"), contains("belt"),contains("dumbbell"))
test <- select(test,  contains("arm"), contains("belt"),contains("dumbbell"))
sum(is.na(train))
```

```
## [1] 1287472
```

4. Clean data There are many column having missing values. Delete columns having NAs

```
nacols <- colSums(is.na(test)) == 0
train <- train[, nacols]
test = test[, nacols]
sum(is.na(train))       # Verify if there are any other missing values
```

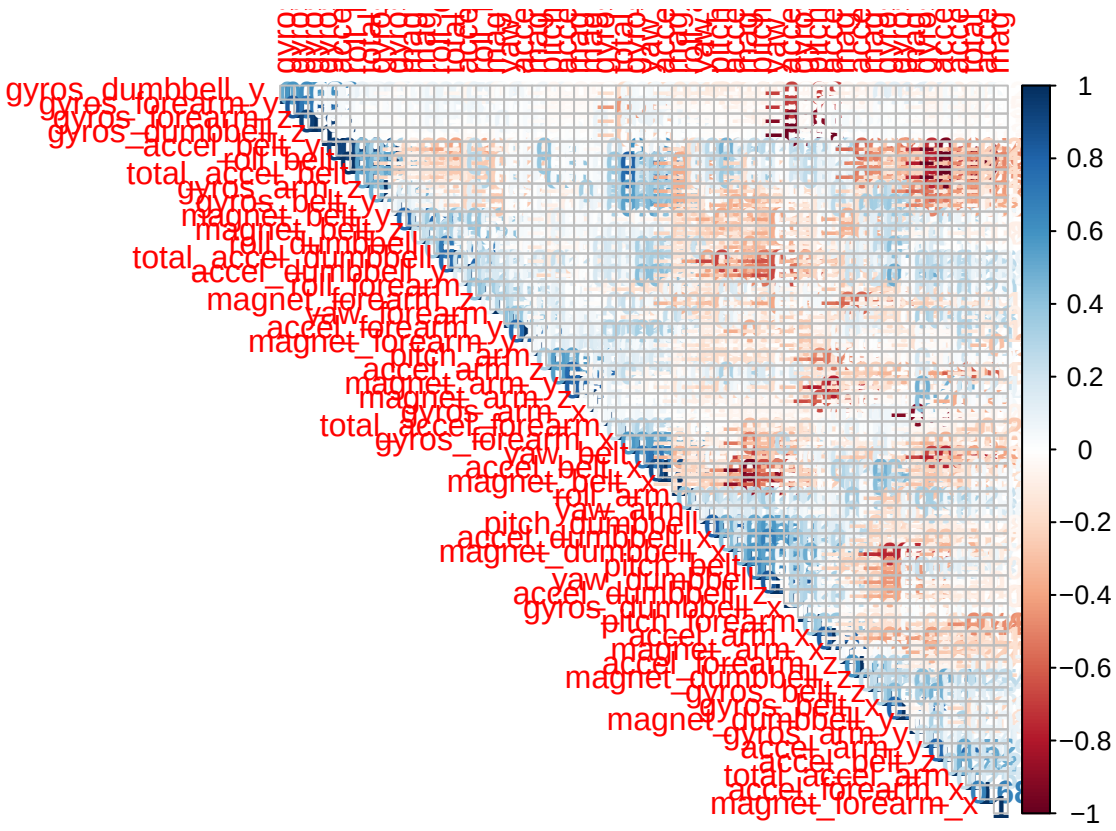
```
## [1] 0
```

```
dim(train)
```

```
## [1] 19622    52
```

5. Plot of correlation matrix between numerical variables

```
corMatrix <- cor(train[sapply(train, is.numeric)])
corrplot::corrplot(corMatrix, method="number", type="upper", order="hclust")
```



6. t-SNE plot
   t-Distributed Stochastic Neighbor Embedding (t-SNE) is a (prize-winning) technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets into low dimensional plot. We will use multidimensional reduction into 2D plane.

```
tsne = Rtsne(as.matrix(train), check_duplicates=FALSE, pca=TRUE,
             perplexity=40, theta=0.5, dims=2)
tsn_embedding = as.data.frame(tsne$Y)
tsn_embedding$Class = target
g = ggplot(tsn_embedding, aes(V1, V2, color=Class)) +
  geom_point() + xlab("") + ylab("") +
  ggtitle("t-SNE Embedding of 'Classe' Outcomes") +
  theme(plot.title = element_text(lineheight=.8,hjust = 0.5))
print(g)
```

In the tSNE plot there is no clear separation of clustering of the 5 levels of Classe outcomes. NOTHING TO SEE HERE! MOVE ON to BUILD PREDICTIVE MODELS.

Build xgb Models

1. Converte data into xgb format for xgboost model

```
y = as.matrix(as.integer(target)-1)
dtrain <- xgb.DMatrix(data=as.matrix(train), label=y)
dtest <- xgb.DMatrix(data=as.matrix(test))
```

2. Setup Paramaeters for CV training
   We will use a 5-fold cross validation with 1000 epochs to achieve the error rate of less than 0.1% for a good classification.

```
param <- list(booster="gbtree",              # tree based boosting
              objective="multi:softprob",    # multiclass classification
              eval_metric="mlogloss",        # evaluation metric
              nthread=13,                    # number of threads to be used
              num_class=5,                   # number of classes
              eta = .03,                     # step size shrinkage
              gamma = 1,                     # minimum loss reduction
              max_depth = 4,                 # maximum depth of tree
              min_child_weight = 4,          # minimum sum of instance weight needed in a child
              subsample = .7,                # part of data instances to grow tree
              colsample_bytree = .5          # subsample ratio of columns when constructing each tree
)
```

3. Estimate number of iteration needed and elapsed time to achieve the minimum error rate

```
set.seed(1230)
tme <- Sys.time()
xgb2cv <- xgb.cv(data = dtrain,
                 params = param,
                 nrounds = 1000,
                 maximize=FALSE,
                 prediction = TRUE,
                 nfold = 5,
                 print_every_n = 200,
                 early_stopping_round=200)
```

```
## [1]  train-mlogloss:1.579544+0.002737    test-mlogloss:1.580287+0.002554
## Multiple eval metrics are present. Will use test_mlogloss for early stopping.
## Will train until test_mlogloss hasn't improved in 200 rounds.
##
## [201]    train-mlogloss:0.289521+0.001460    test-mlogloss:0.311140+0.005060
## [401]    train-mlogloss:0.124548+0.000936    test-mlogloss:0.146553+0.002916
## [601]    train-mlogloss:0.064141+0.000754    test-mlogloss:0.084344+0.002087
## [801]    train-mlogloss:0.037733+0.000485    test-mlogloss:0.055759+0.001585
## [1000]   train-mlogloss:0.027513+0.000338    test-mlogloss:0.044162+0.001500
```

```
elapsedtme <- Sys.time() - tme
```

The elapsed time for 1000 iterations is: 6.5533362 and the mlogloss error is: 0.044162+0.001500

6. Fit the XGBoost gradient boosting model on all of the training data

```
bst <-  xgb.train(data = dtrain,
                  params = param,
                  nrounds = xgb2cv$best_ntreelimit
)
```

7. Calculate mlogloss error for the training data

```
logLoss = function(pred, actual){
  predsums <- 0
  for (i in 1:ncol(pred)){
    predsums <- predsums -1*mean(log(pred[model.matrix(~ actual + 0) - pred[,i] > 0, i]))
  }
  predsums
}
minmax <- function (x) {
  min(max(x, 1E-15), 1-1E-15)
}
pred <- predict(bst, dtrain)
pred <- t(matrix(pred, nrow=5, ncol=nrow(dtrain)))
pred.val <- max.col(pred, "last")
pred = as.data.table(pred)
pred <- mutate_all(pred, minmax)
ll <- logLoss(pred, y)
```

The Error achieved: 0.002839 logloss error

Prediction Results

1. Calculation of confusion matrix

```
confusionMatrix(factor(y+1), factor(pred.val))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2    3    4    5
##          1 5580    0    0    0    0
##          2    0 3797    0    0    0
##          3    0    3 3416    3    0
##          4    0    0    4 3212    0
##          5    0    0    0    1 3606
##
## Overall Statistics
##
##                Accuracy : 0.9994
##                  95% CI : (0.999, 0.9997)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9993
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            1.0000   0.9992   0.9988   0.9988   1.0000
## Specificity            1.0000   1.0000   0.9996   0.9998   0.9999
## Pos Pred Value         1.0000   1.0000   0.9982   0.9988   0.9997
## Neg Pred Value         1.0000   0.9998   0.9998   0.9998   1.0000
## Prevalence             0.2844   0.1937   0.1743   0.1639   0.1838
## Detection Rate         0.2844   0.1935   0.1741   0.1637   0.1838
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   0.9996   0.9992   0.9993   1.0000
```

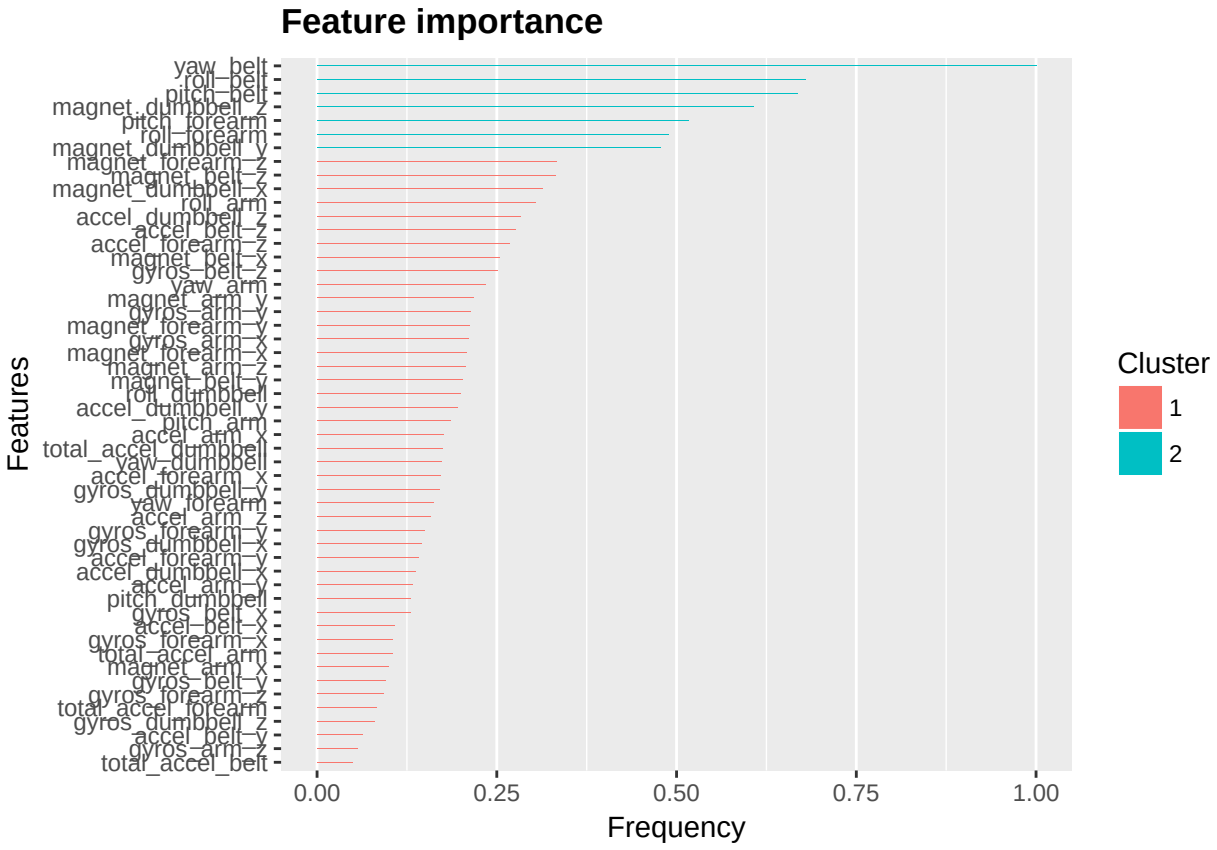
The average accuracy is 99.94%, with error rate of 0.06%. So, expected error rate of less than 0.1% is fulfilled.

2. Predict the test data

```
tPreds <- t(matrix(predict(bst, dtest), nrow=5, ncol=nrow(dtest)))
pred.val <- max.col(tPreds, "last")
classe.val <- toupper(letters[pred.val])
fwrite(data.table(problem_id = rownames(test), classe.val),
       "E:\\Coursera\\Practical_Machine_Learning\\output\\answer.csv")
```

3. identify Important Features

```
model <- xgb.dump(bst, with_stats=TRUE)
names <- names(train)
importance_matrix <- xgb.importance(names, model=bst)
gg <- xgb.ggplot.importance(importance_matrix, measure = "Frequency", rel_to_first = TRUE)
gg + ggplot2::ylab("Frequency")
```


Feature importance
yaw_belt
roll_belt
pitch_belt
magnet_dumbbell_z
pitch_forearm
roll_forearm
magnet_dumbbell_y
magnet_forearm_z
magnet_belt_z
magnet_dumbbell_x
roll_arm
accel_dumbbell_z
accel_belt_z
accel_forearm_z
magnet_belt_x
gyros_belt_z
yaw_arm
magnet_arm_y
gyros_arm_y
magnet_forearm_y
gyros_arm_x
magnet_forearm_x
magnet_arm_z
magnet_belt_y
roll_dumbbell
accel_dumbbell_y
pitch_arm
accel_arm_x
total_accel_dumbbell
yaw_dumbbell
accel_forearm_x
gyros_dumbbell_y
yaw_forearm
accel_arm_z
gyros_forearm_y
gyros_dumbbell_x
accel_forearm_y
accel_dumbbell_x
accel_arm_y
pitch_dumbbell
gyros_belt_x
accel_belt_x
gyros_forearm_x
total_accel_arm
magnet_arm_x
gyros_belt_y
gyros_forearm_z
total_accel_forearm
gyros_dumbbell_z
accel_belt_y
gyros_arm_z
total_accel_belt
Features
0.00
0.25
0.50
0.75
1.00
Frequency
Cluster
1
2