# Coursera - Practical Machine Learning Class Project

*Max Gribov*

*April 15, 2018*

## Summary

This report uses data from accelerometers placed on a dumbell and on the participant to assess if the participant is performing a dumbell lift correctly. The data comes from http://groupware.les.inf.puc-rio.br/har and contains observed accelerometer values. Participants were asked to perform the exercise either correctly, or incorrectly in one of 4 specific ways. This is the "classe" variable in the data set. The report builds several different models on this data to predict "classe" to let the participant know if they are performing the exercise correctly.

The models considered (using caret package):

- Decision Trees using Stochastic Gradient Boosting (gbm)
- Decision Trees using C5.0 algorithm (c50)
- Decision Trees using recursive partitioning (rpart)
- Linear Discriminant Analysis (lda)
- K Nearest Neighors (knn)
- Naive Bayes (nb)
- Regularized Logistic Regression (regLogistic)

Best model in each model class was selected by using 10 fold Cross Validation. Model performance was evaluated by using the confusion matrix to get the prediction accuracy and Cohen's kappa coefficient. All models were trained using parallel processing to take advantage of multi-core systems.

According to my findings, best model was Decision Trees using C5.0 algorithm (C5.0), with Accuracy of 99% and Kappa value of 0.997. This model took about 8 hours to train on my machine.

Runner up was Decision Trees using Stochastic Gradient Boosting (gbm), with Accuracy at 97% and Kappa value of 0.958. This model took about 6 hours to train on my machine.

Third best was K Nearest Neighbors (knn) with Accuracy at 92% and Kappa value of 0.902. This model was much faster to train (minutes), and had high accuracy and Kappa values, however prediction time is slow with this approach.

**NOTE:** Random Forest is expected to have highest performance for a classification task such as this one, but I was not able to train a random forest with k-fold cross validation. I attempted to use parallelizing approach on both windows and linux machines, and each training attempt failed, after running for over 8 hours. Without parallelizing, training would go one for even longer than 8 hours, and I would just stop it since its taking so long.

## Exploring and cleaning data

Original data is a CSV file with 19622 observations, and 160 variables. A lot of the variables are NA, and some of the variables are administrative, such as record ID, so first task is to clean up the data.

```r
library(caret)
```

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

```
set.seed(12345)

# read the csv, make sure to mark empty cols as NA
dat = read.csv("~/code/jh-datasci-projects/practical_machine_learning/pml-training.csv", na.strings=c("

# clean up data to have only meaningful columns, first 8 are admin stuff
clean <- dat[,8:160]

# now deal with large num of NA by using only cols without NA (reduce predictors to 53)
clean <- clean[, colnames(clean)[colSums(is.na(clean)) == 0]]
```

Additionally, “classe” variable with value “A” is over-represented in the data, so remove some of the observations to have roughly same class representation. Then, split up data into training and testing sets.

```
# use mean of other classes: 3510
summary(clean$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
# reduce instance of the class to mean of counts for other classes
classeA <- clean[which(clean$classe == 'A'), ]
classeA <- classeA[0:3510, ]
clean <- rbind(classeA, clean[which(clean$classe != 'A'), ])

# split the original training set into training and testing sets
inTrain = createDataPartition(clean$classe, p = 3/4)[[1]]
training = clean[ inTrain,]
testing = clean[-inTrain,]
```

## Model Training

We will use 10-fold cross validation when selecting the best model by using trainControl() function.

```
fitControl <- trainControl(
  allowParallel = TRUE,
  ## 10-fold CV
  method = "repeatedcv",
  number = 10,
  ## repeated ten times
  repeats = 10
)
```

Each model is trained and analyzed separately, such as the C5.0 example below.

```
# set up parallel processing

# windows
# per https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance
#library(parallel)
#library(doParallel)
#cluster <- makeCluster(detectCores() - 1, outfile="") # convention to leave 1 core for OS
#registerDoParallel(cluster)
```

```r
# linux
library(doMC)
registerDoMC(cores = 3)

test_model <- train(
  classe ~ .,
  data=training,
  method="C5.0", # method would change, for example gbm, rf
  metric="Kappa",
  trControl=fitControl,
  na.action = na.omit
)

# generate the predictions with this model
test_model_pred <- predict(test_model, testing)

# see the accuracy and kappa values for comparison
confusionMatrix(test_model_pred, testing$classe)$overall[["Accuracy"]]
confusionMatrix(test_model_pred, testing$classe)$overall[["Kappa"]]

# save the model object for later use to avoid re-training
saveRDS(test_model, "~/accel_model_c50.rds")
```

I will omit the training code for each of the considered models, and instead load the saved objects directly, and compare their accuracy and kappa values.

```r
# paths need to match whatever was used with saveRDS()
models <- c(
  "accel_model_c50.rds",
  "accel_model_gbm.rds",
  "accel_model_knn.rds",
  "accel_model_lda.rds",
  "accel_model_nb.rds",
  "accel_model_regLogistic.rds",
  "accel_model_rpart.rds"
)

# store the results in a df
results <- data.frame(model_name=double(), accuracy = double(), kappa = double())

# loop through the list of models, load, predict, and measure each one
for (m in models) {
  test_model <- readRDS(paste('/home/max/code/jh-datasci-projects/practical_machine_learning/', m, sep=
  test_model_pred <- predict(test_model, testing)

  acc <- confusionMatrix(test_model_pred, testing$classe)$overall[["Accuracy"]]
  kappa <- confusionMatrix(test_model_pred, testing$classe)$overall[["Kappa"]]

  results[nrow(results) + 1, ] = c(m, acc, kappa)
}

results
```

```
##                      model_name          accuracy             kappa
## 1          accel_model_c50.rds 0.998176014591883 0.997718239813984
## 2         accel_model_gbm.rds 0.967168262653899  0.95893319576429
## 3         accel_model_knn.rds 0.922480620155039 0.903069271566785
## 4         accel_model_lda.rds 0.697218422252622 0.621620199474468
## 5          accel_model_nb.rds 0.753305973552212 0.691814183185029
## 6 accel_model_regLogistic.rds 0.720018239854081 0.649874264353509
## 7       accel_model_rpart.rds 0.389648882808938  0.22344860317837
```