# Random Forest Prediciton of Weight Lifting Performance

## Summary

This report describes the process that was used to select a random forest model, which can predict the manner in which human participants preformed a dumbbell weight lifting exercise.

The data were taken from Weight Lifting Exercise Dataset provided by Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013. http://groupware.les.inf.puc-rio.br/har#ixzz52kKyIwix

Participants performed 5 variations of a weight lifting exercise while being monitored by sensors on the arm, forearm, belt and dumbbell.

The current analysis used a subset of variables collected from the sensors, which provided acceleration, gyroscope and magnetometer measurements, and additional calculated "Euler angles" features (roll, pitch, yaw).

These data were split into a training and validation set, and feature selection methods were performed.

The models provided a high degree of accuracy in regards to predicting the outcomes of the validation set.

The final model that was selected used 9 predictor variables and it yielded ~ 0.989 accuracy as judged by K-fold cross validation methods and by prediction of the validation set outcomes.

## Load data, clean data and remove incomplete variables

The following code loads and cleans the data. Eight columns of data were removed because they were not based on sensor-detected data. There were also 96 variables with a high degree of incomplete data (in which $< 3\%$ of the rows contained data). These derived features were also excluded from the analysis. The resulting training set has 52 predictors. These data were split into "training" and "validation" sets.

```
# load original data file
training0 <- read.table(file = "pml-training.csv", header = T, sep = ",",stringsAsFactors = F)
# replace #DIV/0 entries with NA
training0[training0 == "#DIV/0!"] <- NA
# write temp file
write.csv(training0, file = "train_data_clean")
# read temp file, which is a cleaner version of training data
training1 <- read.csv("train_data_clean")
# remove columns not used in analysis##############################################
# remove first 8 columns (non-sensor data)
training2 <- training1[,-c(1:8)]
# examine data completeness
# there was a high degree of missing data is some of the variables.
na_count <-sapply(training2, function(y) sum(length(which(is.na(y)))))
na_countDF <- data.frame(na_count)
# filter training data for columns with complete data
na_countDF$variable <- row.names(na_countDF)
row.names(na_countDF) <- NULL
na_countDF2 <- na_countDF[na_countDF$na_count <= 0.97*dim(training2)[1],]
keep0 <- na_countDF2$variable
```

```
training3 <-training2[,which(names(training2) %in% keep0)]
# filter testing data for the same columns
testing0 <- read.csv("pml-testing.csv")
# add "problem_id" to keep list
keep_test1 <- c(keep0, "problem_id")
test0 <- testing0[,which(names(testing0) %in% keep_test1)]
#split training3 into training and validation sets
library(caret)
set.seed(666)
inTrain <- createDataPartition(y = training3$classe,p = 0.7, list = F)
train0 <- training3[inTrain,]
val0 <- training3[-inTrain,]
missing_data <- hist(na_countDF$na_count, xlab = "Number of missing values",col = "red",
                     main = "Number of missing values", ylab = "Number of variables")
```


### Model 1: Random forest model based on all 52 predicitors.

Visual inspection of the data with plotting methods suggested the data was very likely not linear in many examples. Therefore, I evaluated the performance of random forest models in their capacity to predict the test set outcomes. Model 1 used all 52 of the predictors that were included in the analysis.

```
# Model 1: 52-predictor model

# define mtry
mtryGrid <- expand.grid(mtry = seq(2,25,4))#
```

```
# set seeds for resampling
set.seed(1)
seeds <- vector(mode = "list", length = 26) # length is = (nresampling)+1
for(i in 1:25) seeds[[i]]<- sample.int(n=1000, 6) # 6 = # of mtry possibilities
seeds[[26]] <- 1 # for the final model
remove(i)
# initiate parallel processing of cpu's 4 processors
library(doParallel)
cl = makeCluster(4)
registerDoParallel(cl)
# run random forest model
RF52 <- train(train0[, -53], train0$classe, method="rf", ntree=300,
              metric="Accuracy", importance=TRUE, na.action=na.omit,
              trControl=trainControl(method="boot", number= 25, seeds=seeds),
              tuneGrid = mtryGrid, allowParallel=TRUE)
# turn off parallel processing
stopCluster(cl)
remove(cl)
registerDoSEQ()
# predict with Model 1 (52-predictor model)
pred_52 <- predict(RF52, newdata = val0)
```

## Summary of Model 1

The variables that were included as predictors.

```
##  [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"
```

Confusion matrix in relation to the validation set

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    8    0    0    0
##          B    0 1125    3    0    0
##          C    0    6 1023    9    1
```

```
##          D    0    0    0  954    7
##          E    0    0    0    1 1074
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9917, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9877   0.9971   0.9896   0.9926
## Specificity            0.9981   0.9994   0.9967   0.9986   0.9998
## Pos Pred Value         0.9952   0.9973   0.9846   0.9927   0.9991
## Neg Pred Value         1.0000   0.9971   0.9994   0.9980   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1912   0.1738   0.1621   0.1825
## Detection Prevalence   0.2858   0.1917   0.1766   0.1633   0.1827
## Balanced Accuracy      0.9991   0.9935   0.9969   0.9941   0.9962
```

Thus, Model 1 was highly accurate. Indeed, the model displayed $> 99.4\%$ overall accuracy with predicting the validation set outcomes. However, the analysis was also computer-resource intensive. Model 2 will use a feature selection method to simplify the model.

## Model 2: Random forest model based on 9 predicitors

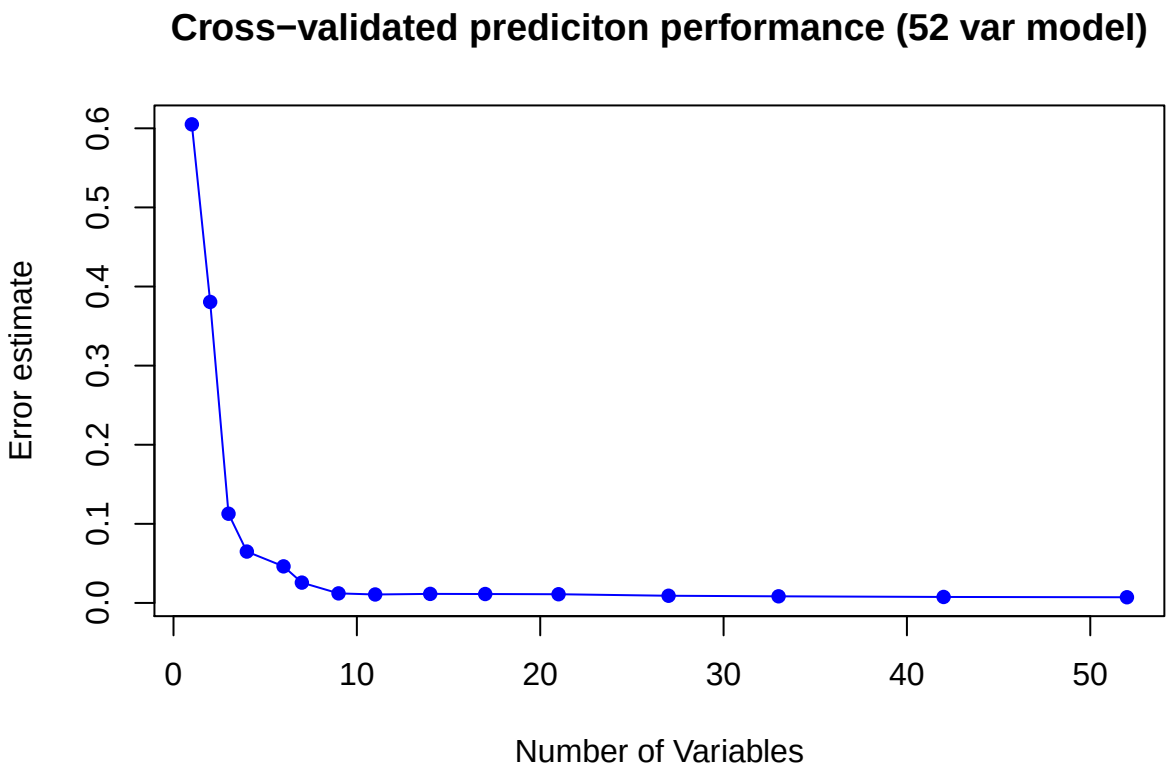
Features were selected with a prediction performance procedure that sequentially reduces the number of predictors using variable importance in a nested cross-validation procedure. Specifically, the rfcv() function for the randomForest package was used in combination with varImp() function for the caret package to identify the 9-predictor model.

```
rm(list = setdiff(ls(), c("train0","val0", "test0", "RF52")))

# Model 2: 9-predictor model based on cross validation performance

# Cross validation of prediction performance
library(randomForest)
rfcv_rf52 <- rfcv(train0[,-53],train0$classe, step=0.8)

plot(rfcv_rf52$n.var,rfcv_rf52$error.cv, col = "blue", pch = 16,
     main = "Cross-validated prediciton performance (52 var model)",
     xlab = "Number of Variables", ylab = "Error estimate")
lines(rfcv_rf52$n.var,rfcv_rf52$error.cv, col = "blue")
```




By virtue of their similar error estimates, the above graph suggests that a 9-predictor model performs almost as well as models with $> 9$ predictors. The varImp() was then used to identify the 9 most important variables. The variables were ranked by the maximum importance value contained within the 5 importance scores that were provided for each of the five classes for each predictor.

```
# filter data for 9 most important variables
viDF <- varImp(RF52)$importance
viDF$max <- apply(viDF, 1, max)
viDF2 <-viDF[order(-viDF$max),]
viDF3 <-viDF2[1:9,]
viDF3$variables <- row.names(viDF3)
keep1 <- c(viDF3$variables, "classe")
train_9cv <-train0[,which(names(train0) %in% keep1)]
val_9cv <-val0[,which(names(val0) %in% keep1)]
# define mtry to examine with 9cv model
mtryGrid <- expand.grid(mtry = c(1:9))
# set seed for resampling
set.seed(1)
seeds <- vector(mode = "list", length = 26) # length is = (nresampling)+1
for(i in 1:25) seeds[[i]]<- sample.int(n=1000, 9) # 9 = # of mtry possibilities
seeds[[26]] <- 1 # for the final model
remove(i)
# initiate parallel processing of cpu's 4 processors
library(doParallel)
cl = makeCluster(4)
registerDoParallel(cl)
# run random forest model
```

```
RF9cv <- train(train_9cv[, -10], train_9cv$classe, method="rf", ntree=300,
               metric="Accuracy",importance=TRUE,na.action=na.omit,
               trControl=trainControl(method="boot", number= 25, seeds=seeds),
               tuneGrid = mtryGrid,allowParallel=TRUE)
# turn off parallel processing
stopCluster(cl)
remove(cl)
registerDoSEQ()
# predict with Model 2 (9 variable model based on cross validation performance)
pred_9cv <- predict(RF9cv, newdata = val_9cv)
```

## Summary of Model 2

The variables that were included as predictors.

```
## [1] "roll_belt"         "pitch_belt"        "yaw_belt"
## [4] "gyros_belt_z"      "magnet_dumbbell_y" "magnet_dumbbell_z"
## [7] "roll_forearm"      "pitch_forearm"     "accel_forearm_x"
```

Confusion matrix in relation to the validation set

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1665    8    0    0    0
##          B    7 1109    3    2    3
##          C    1   21 1020    5    2
##          D    1    1    3  955    6
##          E    0    0    0    2 1071
##
## Overall Statistics
##
##                Accuracy : 0.989
##                  95% CI : (0.9859, 0.9915)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.986
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9946   0.9737   0.9942   0.9907   0.9898
## Specificity            0.9981   0.9968   0.9940   0.9978   0.9996
## Pos Pred Value         0.9952   0.9867   0.9724   0.9886   0.9981
## Neg Pred Value         0.9979   0.9937   0.9988   0.9982   0.9977
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2829   0.1884   0.1733   0.1623   0.1820
## Detection Prevalence   0.2843   0.1910   0.1782   0.1641   0.1823
## Balanced Accuracy      0.9964   0.9853   0.9941   0.9942   0.9947
```

The overall accuracy of Model 2 was ~ 98.9%. The results indicate that the 9-predictor model is almost

as accurate as the 52-predictor model. The 9-predictor model was selected as the final model because it predicted with high accuracy and it is relativity simple compared to the others tested.

## K-Fold Cross Validation

K-fold cross validation was used to estimate how well the training data might predict the test set.

```
# k-fold cross validation
mtryGrid <- expand.grid(mtry = c(3))#
set.seed(123)
seeds <- vector(mode = "list", length = 51) # length is = (nresampling)+1
for(i in 1:50) seeds[[i]]<- sample.int(n=1000, 1) # number of tuning parameters
seeds[[51]] <- 1 # for the final model
remove(i)
# define training control
train_control <- trainControl(method="repeatedcv", number=10, repeats=5,
                              seeds = seeds)
# initiate parallel processing
library(doParallel)
cl = makeCluster(4)
registerDoParallel(cl)

# train the model
kfold_model <- train(classe ~., data=train_9cv, tuneGrid = mtryGrid,
                     trControl=train_control, method="rf", allowParallel=TRUE)
# stop parallel processing
stopCluster(cl)
remove(cl)
registerDoSEQ()
```

The summary of the K-fold cross validaton procedure.

```
## Random Forest
##
## 13737 samples
##     9 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 12363, 12362, 12363, 12363, 12363, 12363, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9889493  0.986024
##
## Tuning parameter 'mtry' was held constant at a value of 3
```

The average accuracy across 5 repetitions of 10-fold cross validation was 0.989. This suggests the models will correctly classify close to 99% of the test data. Notably, this value is very similar to the the error estimate provided by prediction of the validation set. Thus, its plausible to speculate that this model could correctly identify 19 or 20 of the test samples.