PredictionAssignment

ThinkersPark

2023-07-09

This report presents the Practical Machine Learning - Prediction Assignment, part of Coursera Data Science with R specialisation, by Johns Hopkins University.

1. Executive Summary

In this assignment, data is analysed from accelerometers on the belt, forearm, arm, and dumbell of 6 participants performing barbell lifts, in 5 different ways; Data courtesy of http://groupware.les.inf.puc-rio.br/har.

The objective of the assignment is to build a prediction model, which - based on accelerometer measurements - would predict the manner in which the exercise is performed.

This report organised as follows: Section 2 contains a general overview of the data and basic exploratory analysis; Section 3 - discussion of model fit, accuracy and performance; Section 4 - interpretations and adjustments; Section 5 - cross-validation; Section 6 concludes.

2. Data overview and pre-processing

Loading the necessary packages and reading in data sets (downloaded to the ./Data folder via link provided above):

```r
library(ggplot2);library(caret);library(randomForest);library(gbm);library(forecast);
library(fpc);library(dplyr);library(ggpubr)
training <- read.csv("./Data/pml-training.csv")
testing <- read.csv("./Data/pml-testing.csv")
## head(training); dim(training); dim(testing); colnames(training); colnames(testing)
```

A quick look at the data makes a few points clear (code commented out to save a bit of space):

(1) The first column of both the training and the testing set is just the observation index, and will be removed.

(2) "Classe" is originally a character variable, will be converted into a factor variable for ease of handling; The same applies to "user_name".

(3) The testing set does not have the "classe" variable, meaning there is no actual (supervised) information about the manner in which the exercise was performed. In this case, the original testing dataset will be converted to an (unsupervised) validation set, and the original training dataset will be partitioned into (supervised) training and testing sets.

(4) There are NAs across the datasets. Removing NAs needs to be handled with care, as - in theory - a given measurement being an "NA" may be a good predictor of "classe" (in other words, certain measurement may not be missing at random).

(a) Examine presence/ frequency of NAs across measurements for each "classe" value, to evaluate NAs pattern,
(b) Depending on the outcome of the above, remove variables with excess proportion of NAs.

(5) There are multiple variables in the set, it makes sense to reduce the dimensionality:

(a) Remove those with little variability,
(b) Remove those which are highly correlated with each other to avoid overfit/ reduce the computational load.

Steps (1), (2) and (3):

```
training <- training[,-1]; testing <- testing[,-1]
training$classe <- factor(training$classe);
training$user_name <- factor(training$user_name);
testing$user_name <- factor(testing$user_name);
set.seed(12345)
validation <- testing
inTrain <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
testing <- training[-inTrain,]
training <- training[inTrain,]
## dim(training); dim(testing); dim(validation)
```

Step (4a) - checking that for each variable, the proportion of NAs is the same for each level of "classe" (printing "S" commented out to save a bit of space, when un-commented it show that for reach variable with NAs present, NAs occur in exactly the same proportion of 98% for each level of "classe"):

```
propnas <- function(x) {round(length(x[which(is.na(x)==TRUE)])/ length(x),2)}
S <- training %>% group_by(classe) %>% summarise_all("propnas")
## print(as.data.frame(S))
```

Step (4b) - removing variables with more than 75% of NAs (indiscriminately across all levels of "classe", based on the earlier analysis); The threshold is a matter of choice, in this case any threshold <= 98% will have the same effect:

```
s <- which(sapply(training,propnas)>0.75)
training <- training[,-s]; testing <- testing[,-s]; validation <- validation[,-s]
## dim(training); dim(testing); dim(validation)
```

Step (5a) and (5b) - removing variables with low variability and those highly correlated with each other (correlation cut-off point set at 0.75, again a matter of choice):

```
nearzeros <- nearZeroVar(training[,-length(training[1,])],saveMetrics=TRUE)
training <- training[,c(which(nearzeros$nzv==FALSE),length(training[1,]))]
testing <- testing[,c(which(nearzeros$nzv==FALSE),length(testing[1,]))]
validation <- validation[,which(nearzeros$nzv==FALSE)]
## dim(training); dim(testing); dim(validation)
```

```
u <- which(unlist(lapply(training,class))%in%c("numeric","integer"))
f <- findCorrelation(cor(training[,u],use="complete.obs"),cutoff=0.75,
                     verbose=FALSE,names=FALSE,exact=FALSE)
training <- training[,-f]; testing <- testing[,-f]; validation <- validation[,-f]
## dim(training); dim(testing); dim(validation)
```

As a result, we are left with a set of 35 candidate explanatory variables (excl. "classe"), out of the initial 136.

3. Model fit, accuracy and performance

We will compare 3 models suitable for a classification problem: Random Forest model (RF), Generalised Boosted Regression model (GBM), and their ensemble. For each model, the following are calculated:

- Run-time,
- In-sample/ Out-of-sample accuracy (based on the partitioned training/ resp. testing set),
- Out-of-Sample prediction frequency table.

And then the following plots are generated:

- For RF & GBM models, illustration of out-of-bag error stabilising with increasing number of trees,
- For all models, importance of variables,
- For all models, illustration of out-of-sample prediction.

Random Forest (RF) model fit:

```
set.seed(12345)
start.time <- Sys.time()
modFitRf <- train(classe ~ .,data=training,method="rf", ntree=150,
                  prox=TRUE,na.action=na.omit)
end.time <- Sys.time()
time.taken.rf <- end.time - start.time

predRfInSample <- predict(modFitRf,training)
predRfOOSample <- predict(modFitRf,testing)
RfStatInSample <- confusionMatrix(predRfInSample,training$classe)
RfStatOOSample <- confusionMatrix(predRfOOSample,testing$classe)
```

Generalised Boosted Regression (GBM) model fit:

```
set.seed(12345)
start.time <- Sys.time()
modFitGbm <- train(classe ~ .,data=training,method="gbm", verbose=FALSE)
end.time <- Sys.time()
time.taken.gbm <- end.time - start.time

predGbmInSample <- predict(modFitGbm,training)
```

```
predGbmOOSample <- predict(modFitGbm,testing)
GbmStatInSample <- confusionMatrix(predGbmInSample,training$classe)
GbmStatOOSample <- confusionMatrix(predGbmOOSample,testing$classe)
```

RF & GBM Out-of-bag error (stable after 150 trees):

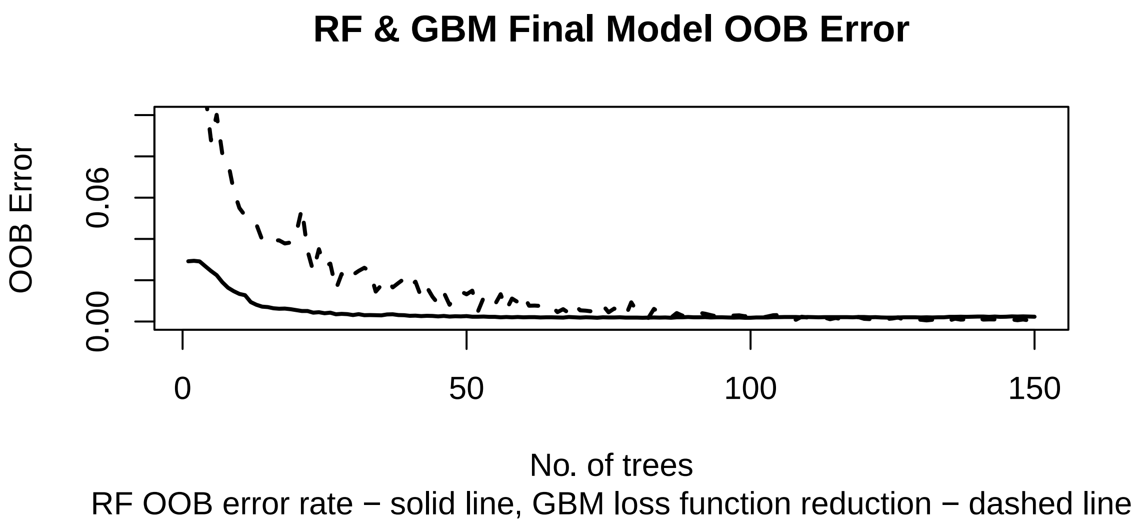

Figure 1: RF & GBM Out-of-bag error.

RF & GBM ensemble model fit:

```
set.seed(12345)
predDF <- data.frame(predRfOOSample,predGbmOOSample,classe=testing$classe)
start.time <- Sys.time()
modFitEns <- train(classe ~.,method="rf",data=predDF)
end.time <- Sys.time()
time.taken.ens <- end.time - start.time

predEns<- predict(modFitEns,predDF)
EnsStat <- confusionMatrix(predEns,testing$classe)
```

Model accuracy and out-of sample prediction:

```
## [1] "Random Forest (RF) model:"

## [1] "Runtime: 58.7200 mins"

## [1] "Accuracy in-sample (training): 1.0000; Kappa: 1.0000"

## [1] "Accuracy out-of-sample (testing): 0.9978; Kappa: 0.9972"

## [1] "Out-of-sample prediction (columns) vs. actual (rows):"
```

```
##
## predRfOOSample    A    B    C    D    E
##              A 1673    0    0    0    0
##              B    1 1139    1    0    0
##              C    0    0 1018    3    0
##              D    0    0    7  960    0
##              E    0    0    0    1 1082

## [1] "Generalised Boosted Regression (GBM) model:"

## [1] "Runtime: 14.0100 mins"

## [1] "Accuracy in-sample (training): 0.9977; Kappa: 0.9971"

## [1] "Accuracy out-of-sample (testing): 0.9941; Kappa: 0.9925"

## [1] "Out-of-sample prediction (columns) vs. actual (rows):"

##
## predGbmOOSample    A    B    C    D    E
##               A 1672    3    0    0    0
##               B    2 1135    2    0    0
##               C    0    1 1015    9    1
##               D    0    0    9  950    3
##               E    0    0    0    5 1078

## [1] "Ensemble RF/ GBM model:"

## [1] "Runtime: 1.1100 mins"

## [1] "Accuracy out-of-sample (testing): 0.9978; Kappa: 0.9972"

## [1] "Out-of-sample prediction (columns) vs. actual (rows):"

##
## predEns    A    B    C    D    E
##       A 1673    0    0    0    0
##       B    1 1139    1    0    0
##       C    0    0 1018    3    0
##       D    0    0    7  960    0
##       E    0    0    0    1 1082
```

Importance of variables:

Out-of-sample prediction - comparison between models:

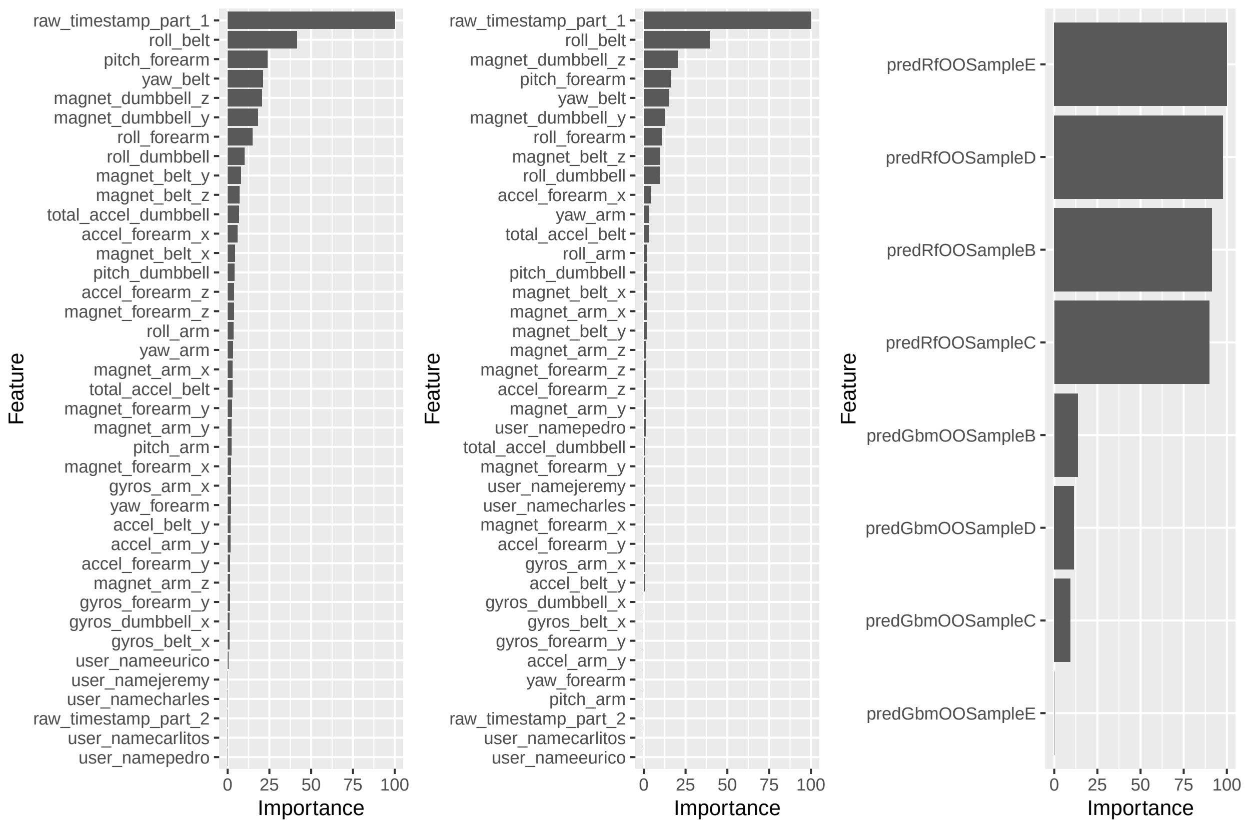

Figure 2: Importance of Variables

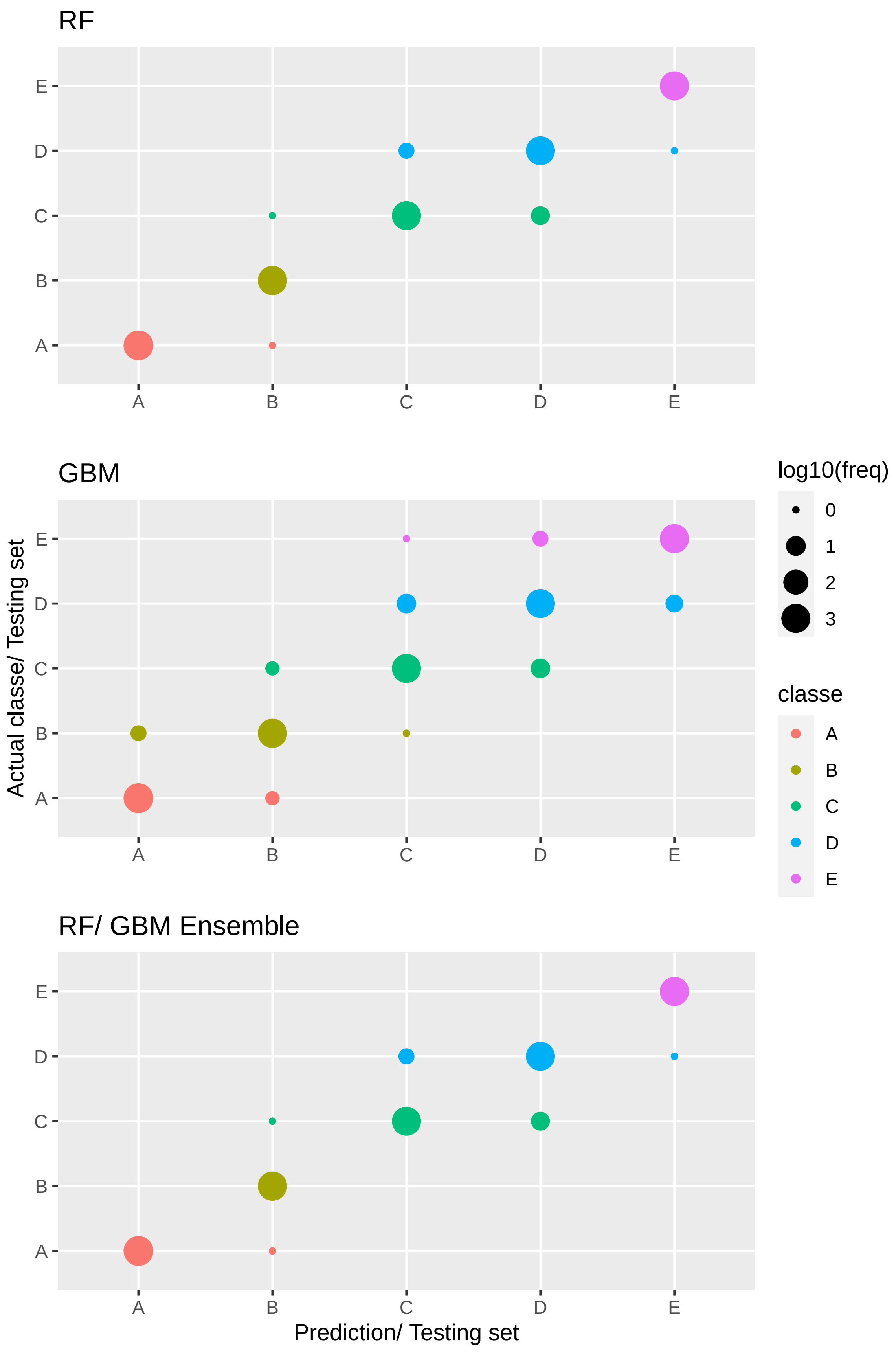

Figure 3: Out-of-sample prediction - comparison between models

Interpretations and adjustments

The above model fit experiments show that:

- Random Forest (RF) model achieves a very high level of out-of sample accuracy (0.997), although is fairly time-consuming to fit (just under 1 hour),
- In comparison, Generalised Boosted Regression (GBM) model is just marginally less accurate out-of sample (0.994), and much faster to fit (approx. 17 mins),
- The ensemble model does not offer real accuracy benefit vs. Random Forest, and primarily relies on Random Forest classifications;

In both RF and GBM models, timestamp variable is an important determinant of classifying decisions, indicating that perhaps the exercises were performed in a specific order. In general, specific users may have also performed only specific types of exercises, athough in this case it is not supported by username variable importance. For the models to be more suitable to a randomised setting - both the timestamp and username variables have been removed, and all models refitted based on the remaining variables (pre-processing steps described in earlier sections continue to apply):

```
training_rmuserts <- training[,-c(1,2,3)]; testing_rmuserts <- testing[,-c(1,2,3)]
validation_rmuserts <- validation[,-c(1,2,3)]
## dim(training_rmuserts); dim(testing_rmuserts); dim(validation_rmuserts)
```

Refit code is included in the appendix, and works exactly the same as the original fitting sequence. Refitted model accuracy and out-of sample prediction:

```
## [1] "Refitted Random Forest (ReRF) model:"

## [1] "Runtime: 49.7100 mins"

## [1] "Accuracy in-sample (training): 1.0000; Kappa: 1.0000"

## [1] "Accuracy out-of-sample (testing): 0.9954; Kappa: 0.9942"

## [1] "Out-of-sample prediction (columns) vs. actual (rows):"

##
## predReRfOOSample    A    B    C    D    E
##                A 1673    1    0    0    0
##                B    1 1138    6    0    0
##                C    0    0 1020   17    1
##                D    0    0    0  946    0
##                E    0    0    0    1 1081

## [1] "Refitted Generalised Boosted Regression (ReGBM) model:"

## [1] "Runtime: 12.1600 mins"

## [1] "Accuracy in-sample (training): 0.9667; Kappa: 0.9578"

## [1] "Accuracy out-of-sample (testing): 0.9521; Kappa: 0.9394"
## [1] "Out-of-sample prediction (columns) vs. actual (rows):"
```

```
##
## predReGbmOOSample    A    B    C    D    E
##                 A 1643   45    0    0    1
##                 B   21 1058   59    4   16
##                 C    2   30  951   38   15
##                 D    8    3   15  918   17
##                 E    0    3    1    4 1033
```

```
## [1] "Refitted Ensemble RF/ GBM model:"
```

```
## [1] "Runtime: 1.2500 mins"
```

```
## [1] "Accuracy out-of-sample (testing): 0.9954; Kappa: 0.9942"
```

```
## [1] "Out-of-sample prediction (columns) vs. actual (rows):"
```

```
##
## predReEns    A    B    C    D    E
##         A 1673    1    0    0    0
##         B    1 1138    6    0    0
##         C    0    0 1020   17    1
##         D    0    0    0  946    0
##         E    0    0    0    1 1081
```

Different variables become important in place of the discarded timestamp and username variables, however at minimal drop of accuracy (~0.003 for the RF and Ensemble models). Looking at the original and the refitted RF, classifications on the testing set are just slightly different.

Out-of-sample prediction - RF vs. ReRF:

Cross-validation

Given the negligible accuracy benefit between RF and Ensemble modes, both RF models (original and refitted) will be applied to the validation set ("unsupervised"/ originally the testing set).Both classify in the same manner as shown by table below:

```
## [1] "Classifications for the validation set (rows - original RF, columns - refitted RF): "
```

```
##          predReRfVal
## predRfVal A B C D E
##         A 7 0 0 0 0
##         B 0 8 0 0 0
##         C 0 0 1 0 0
##         D 0 0 0 1 0
##         E 0 0 0 0 3
```

Without the actual ("supervised") values of the classe variable in the validation set, quantification of the model accuracy is somewhat problematic. To give an indication, certain cluster stats can be calculated for the validation set, similar to those used in unsupervised learning setup. In particular, the Calinski-Harabasz

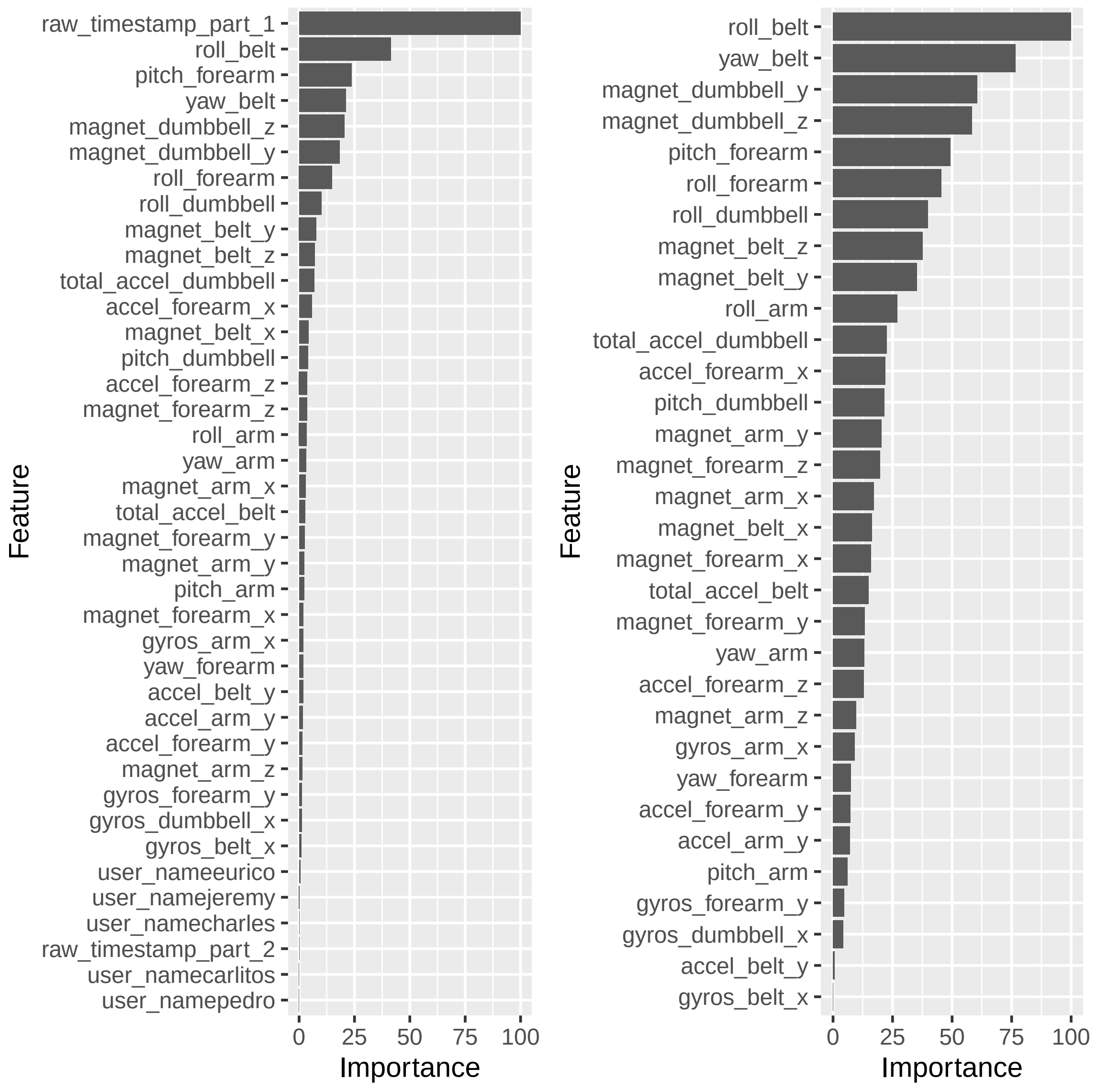

Figure 4: Importance of Variables - original and refitted RF

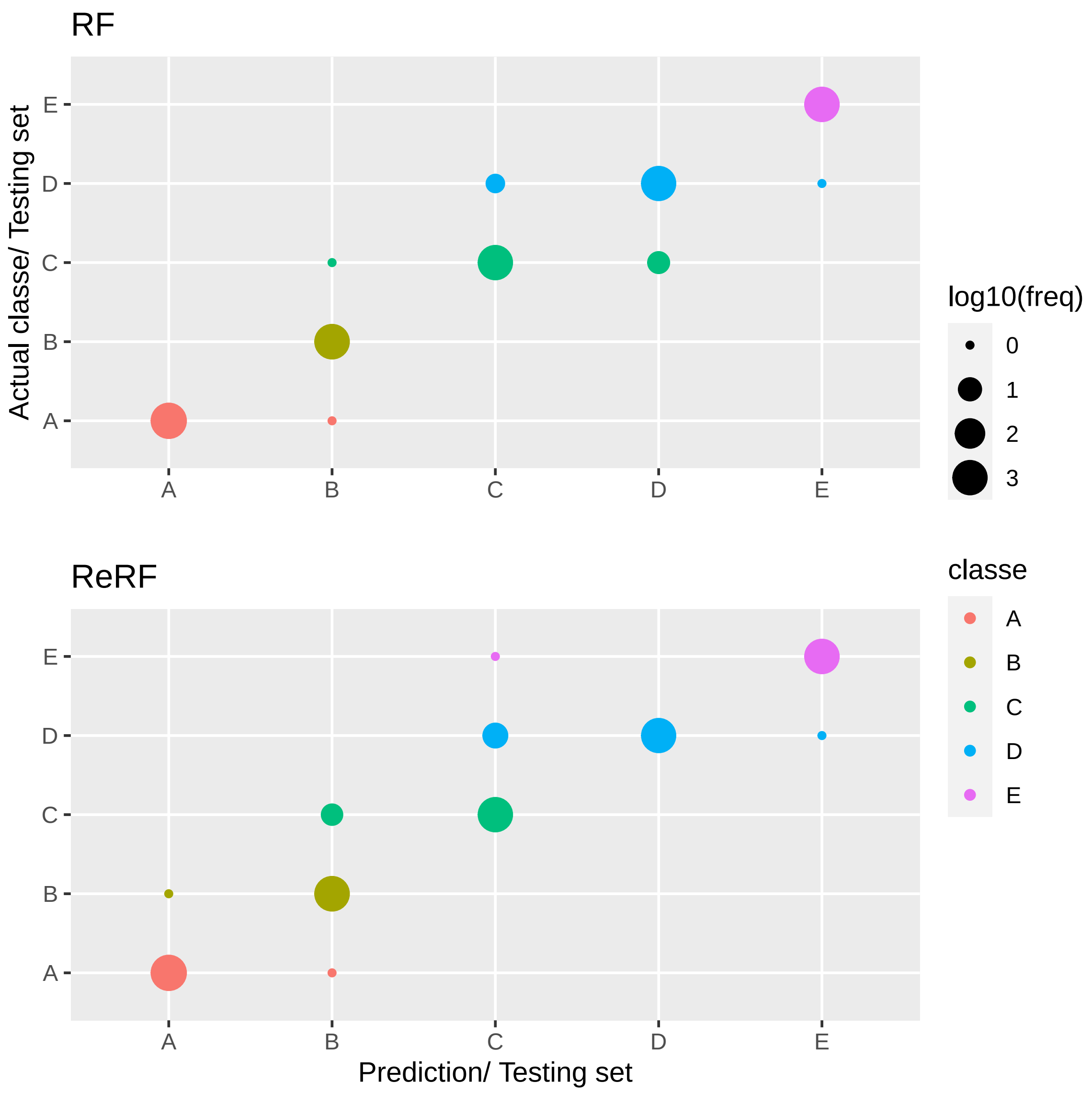

Figure 5: Out-of-sample prediction - RF vs. ReRF.

stat (CH) may be useful as a measure taking into account both the internal dispersion of clusters and the dispersion between clusters. For both RF models the value of this stat is approx. 1 (slightly lower for the original RF), which can be considered quite low (for other tested/ discarded models such as Linear Discriminant Analysis (LDA), or simpler ensemble models, with accuracy approx. 0.4-0.6, the CH stat was in the range of 10-15).

- For an overview of unsupervised learning metrics, please refer to "Evaluation Metrics for Unsupervised Learning Algorithms", Palacio-Nino & Berzal, 2019, available here: https://arxiv.org/pdf/1905.05667.pdf
- For details of R implementation, please refer to https://search.r-project.org/CRAN/refmans/fpc/html/cluster.stats.html

```
## [1] "RF CH stat: 1.04; Refitted RF CH stat: 0.97"
```

Conclusion

The originally fitted RF and GBM models each show a very high in-sample and out-of-sample accuracy, with their ensemble not offering a real accuracy benefit (but adding complexity). However, both original models rely on the stamptime variable as the key predictor of the manner in which the exercise was performed, suggesting the different manners may have been sequenced in the same way.

In practice, applying such models to randomly sequenced exercises may have limited analytical value/ lead to outright incorrect results, and so the models were refitted without the timestamp (and username) variables. This has resulted in just a marginal drop in accuracy, however making the refitted models more suitable to a randomised setting.

The refitted Random Forest (ReRF) model shows out-of-sample accuracy of 0.996 and takes just under 1 hour to fit, post data pre-processing steps. The model considers roll_belt, yaw_belt and magnet_dumbell_y/z as the most important variables, and classifies the validation set with relatively very small internal dispersion of clusters and the dispersion between clusters.

Appendix

Model refit code

RF refit:

```
set.seed(12345)
start.time <- Sys.time()
modReFitRf <- train(classe ~ .,data=training_rmuserts,method="rf", ntree=150,
                  prox=TRUE,na.action=na.omit)
end.time <- Sys.time()
time.taken.rerf <- end.time - start.time

predReRfInSample <- predict(modReFitRf,training_rmuserts)
predReRfOOSample <- predict(modReFitRf,testing_rmuserts)
ReRfStatInSample <- confusionMatrix(predReRfInSample,training_rmuserts$classe)
ReRfStatOOSample <- confusionMatrix(predReRfOOSample,testing_rmuserts$classe)
```

GBM refit:

```
set.seed(12345)
start.time <- Sys.time()
modReFitGbm <- train(classe ~ .,data=training_rmuserts,method="gbm",verbose=FALSE)
end.time <- Sys.time()
time.taken.regbm <- end.time - start.time

predReGbmInSample <- predict(modReFitGbm,training_rmuserts)
predReGbmOOSample <- predict(modReFitGbm,testing_rmuserts)
ReGbmStatInSample <- confusionMatrix(predReGbmInSample,training_rmuserts$classe)
ReGbmStatOOSample <- confusionMatrix(predReGbmOOSample,testing_rmuserts$classe)
```

Ensemble refit:

```
set.seed(12345)
predReDF <- data.frame(predReRfOOSample,predReGbmOOSample,classe=testing_rmuserts$classe)
start.time <- Sys.time()
modReFitEns <- train(classe ~.,method="rf",data=predReDF)
end.time <- Sys.time()
time.taken.reens <- end.time - start.time

predReEns<- predict(modReFitEns,predReDF)
ReEnsStat <- confusionMatrix(predReEns,testing_rmuserts$classe)
```

Plot generation & stat printing code

Figure 1: RF & GBM Out-of-bag error.

```
plot(modFitRf$finalModel$err.rate[,1],main="RF & GBM Final Model OOB Error",
     type='l',lwd=2, xlab="No. of trees", ylab="OOB Error",
     sub="RF OOB error rate - solid line, GBM loss function reduction - dashed line",
     mai=c(5,2,4,2),ylim=c(0,0.1))
lines(modFitGbm$finalModel$oobag.improve,lty=2, lwd=2, col=1)
```

Original model accuracy and out-of sample prediction:

```
print("Random Forest (RF) model:")
sprintf("Runtime: %.4f mins", round(as.numeric(time.taken.rf,units="mins"),2))
sprintf("Accuracy in-sample (training): %.4f; Kappa: %.4f",RfStatInSample$overall[1],RfStatInSample$ove
sprintf("Accuracy out-of-sample (testing): %.4f; Kappa: %.4f",RfStatOOSample$overall[1],RfStatOOSample$
print("Out-of-sample prediction (columns) vs. actual (rows):")
table(predRfOOSample,testing$classe)

print("Generalised Boosted Regression (GBM) model:")
sprintf("Runtime: %.4f mins", round(as.numeric(time.taken.gbm,units="mins"),2))
sprintf("Accuracy in-sample (training): %.4f; Kappa: %.4f",
        GbmStatInSample$overall[1],GbmStatInSample$overall[2])
sprintf("Accuracy out-of-sample (testing): %.4f; Kappa: %.4f",
        GbmStatOOSample$overall[1], GbmStatOOSample$overall[2])
print("Out-of-sample prediction (columns) vs. actual (rows):")
table(predGbmOOSample,testing$classe)
```

```r
print("Ensemble RF/ GBM model:")
sprintf("Runtime: %.4f mins", round(as.numeric(time.taken.ens,units="mins"),2))
sprintf("Accuracy out-of-sample (testing): %.4f; Kappa: %.4f",
        EnsStat$overall[1], EnsStat$overall[2])
print("Out-of-sample prediction (columns) vs. actual (rows):")
table(predEns,testing$classe)
```

Figure 2: Importance of Variables.

```r
g1 <- ggplot(varImp(modFitRf),xlab="", main="RF")
g2 <- ggplot(varImp(modFitGbm), ylab="",  main="GB")
g3 <- ggplot(varImp(modFitEns),xlab="", ylab="", main="RF & GB Ensemble")
ggarrange(g1,g2,g3,ncol=3,nrow=1)
```

Figure 3: Out-of-sample prediction - comparison between models.

```r
freqDataRf <- as.data.frame(table(predRfOOSample,testing$classe));
names(freqDataRf) <- c("predRfOOSample", "classe", "freq");
freqDataGbm <- as.data.frame(table(predGbmOOSample,testing$classe))
names(freqDataGbm) <- c("predGbmOOSample", "classe", "freq")
freqDataEns <- as.data.frame(table(predEns,testing$classe))
names(freqDataEns) <- c("predEns", "classe", "freq")

q1 <- qplot(predRfOOSample,classe,colour=classe,size=log10(freq),
      data=freqDataRf[freqDataRf$freq>0,], main="RF",
      xlab="", ylab="")
q2 <- qplot(predGbmOOSample,classe,colour=classe,size=log10(freq),
            data=freqDataGbm[freqDataGbm$freq>0,], main="GBM",
            xlab="", ylab="Actual classe/ Testing set")
q3 <- qplot(predEns,classe,colour=classe,size=log10(freq), data=freqDataEns[freqDataEns$freq>0,],
            main="RF/ GBM Ensemble",
            xlab="Prediction/ Testing set", ylab="")
ggarrange(q1,q2,q3,ncol=1, nrow=3,common.legend = TRUE,legend="right")
```

Refitted model accuracy and out-of sample prediction:

```r
print("Refitted Random Forest (ReRF) model:")
sprintf("Runtime: %.4f mins", round(as.numeric(time.taken.rerf,units="mins"),2))
sprintf("Accuracy in-sample (training): %.4f; Kappa: %.4f",
        ReRfStatInSample$overall[1],ReRfStatInSample$overall[2])
sprintf("Accuracy out-of-sample (testing): %.4f; Kappa: %.4f",
        ReRfStatOOSample$overall[1], ReRfStatOOSample$overall[2])
print("Out-of-sample prediction (columns) vs. actual (rows):")
table(predReRfOOSample,testing_rmuserts$classe)

print("Refitted Generalised Boosted Regression (ReGBM) model:")
sprintf("Runtime: %.4f mins", round(as.numeric(time.taken.regbm,units="mins"),2))
sprintf("Accuracy in-sample (training): %.4f; Kappa: %.4f",
       ReGbmStatInSample$overall[1],ReGbmStatInSample$overall[2])
sprintf("Accuracy out-of-sample (testing): %.4f; Kappa: %.4f",
        ReGbmStatOOSample$overall[1], ReGbmStatOOSample$overall[2])
print("Out-of-sample prediction (columns) vs. actual (rows):")
```

```
table(predReGbmOOSample,testing_rmuserts$classe)

print("Refitted Ensemble RF/ GBM model:")
sprintf("Runtime: %.4f mins", round(as.numeric(time.taken.reens,units="mins"),2))
sprintf("Accuracy out-of-sample (testing): %.4f; Kappa: %.4f",
        ReEnsStat$overall[1], ReEnsStat$overall[2])
print("Out-of-sample prediction (columns) vs. actual (rows):")
table(predReEns,testing_rmuserts$classe);
```

Figure 4: Importance of Variables - original and refitted RF.

```
g11 <- ggplot(varImp(modFitRf),xlab="", main="RF")
g12 <- ggplot(varImp(modReFitRf),ylab="", main="ReRF")
ggarrange(g11,g12,ncol=2,nrow=1)
```

Figure 5: Out-of-sample prediction - RF vs. ReRF.

```
freqDataReRf <- as.data.frame(table(predReRfOOSample,testing_rmuserts$classe));
names(freqDataReRf) <- c("predReRfOOSample", "classe", "freq");

q11 <- qplot(predRfOOSample,classe,colour=classe,size=log10(freq),
      data=freqDataRf[freqDataRf$freq>0,], main="RF",
      xlab="",ylab="Actual classe/ Testing set")
q12 <- qplot(predReRfOOSample,classe,colour=classe,size=log10(freq),
      data=freqDataReRf[freqDataReRf$freq>0,],main="ReRF",xlab="Prediction/ Testing set",ylab="")
ggarrange(q11,q12,ncol=1,nrow=2,common.legend = TRUE,legend="right")
```

Classifications for the validation set:

```
predRfVal <- predict(modFitRf,validation)
predReRfVal <- predict(modReFitRf,validation_rmuserts)
print("Classifications for the validation set (rows - original RF, columns - refitted RF): ")
table(predRfVal,predReRfVal)
```

CH stats:

```
print("RF CH stat: ");cluster.stats(dist(validation),as.numeric(predRfVal))$ch
print("Refitted RF CH stat: ");cluster.stats(dist(validation_rmuserts),as.numeric(predReRfVal))$ch
```