ML - Project

Dr.Senthil

20/08/2019

Executive Summary

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

Steps to be followed:

1. Data processing
2. Data Exploration

3. Model selection
4. Model examination
5. Conclusion

6. Prediction

Data Processing

Change 'am' to factor (0 = automatic, 1 = manual) Make cylinders a factor as well (since it is not continious)

```
training.raw <- read.csv("pml-training.csv")
testing.raw <- read.csv("pml-testing.csv")
```

Data Exploration

Look at the dimensions & head of the dataset to get an idea

```
dim(training.raw)
```

```
## [1] 9623  160
```

```
# Remove blank values
mNA = 20
maxNACount <- nrow(training.raw) / 100 * mNA
removeColumns <- which(colSums(is.na(training.raw) | training.raw=="") > maxNACount)
training.cleaned01 <- training.raw[,-removeColumns]
testing.cleaned01 <- testing.raw[,-removeColumns]
```

Remove all time related data, since we won't use those

```
removeColumns <- grep("timestamp", names(training.cleaned01))
training.cleaned02 <- training.cleaned01[,-c(1, removeColumns )]
testing.cleaned02 <- testing.cleaned01[,-c(1, removeColumns )]
```

Convert all factors to integers

```
classeLevels <- levels(training.cleaned02$classe)
training.cleaned03 <- data.frame(data.matrix(training.cleaned02))
training.cleaned03$classe <- factor(training.cleaned03$classe, labels=classeLevels)
testing.cleaned03 <- data.frame(data.matrix(testing.cleaned02))
```

Set the dataset to be explored

```
training.cleaned <- training.cleaned03
testing.cleaned <- testing.cleaned03
```

Split the current training in a test and train set to work with

```
set.seed(19791108)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.1
```

```
classeIndex <- which(names(training.cleaned) == "classe")
partition <- createDataPartition(y=training.cleaned$classe, p=0.75, list=FALSE)
training.subSetTrain <- training.cleaned[partition, ]
training.subSetTest <- training.cleaned[-partition, ]
```

Fields that have high correlations

```
correlations <- cor(training.subSetTrain[, -classeIndex], as.numeric(training.subSetTrain$classe))
bestCorrelations <- subset(as.data.frame(as.table(correlations)), abs(Freq)>0.3)
bestCorrelations
```

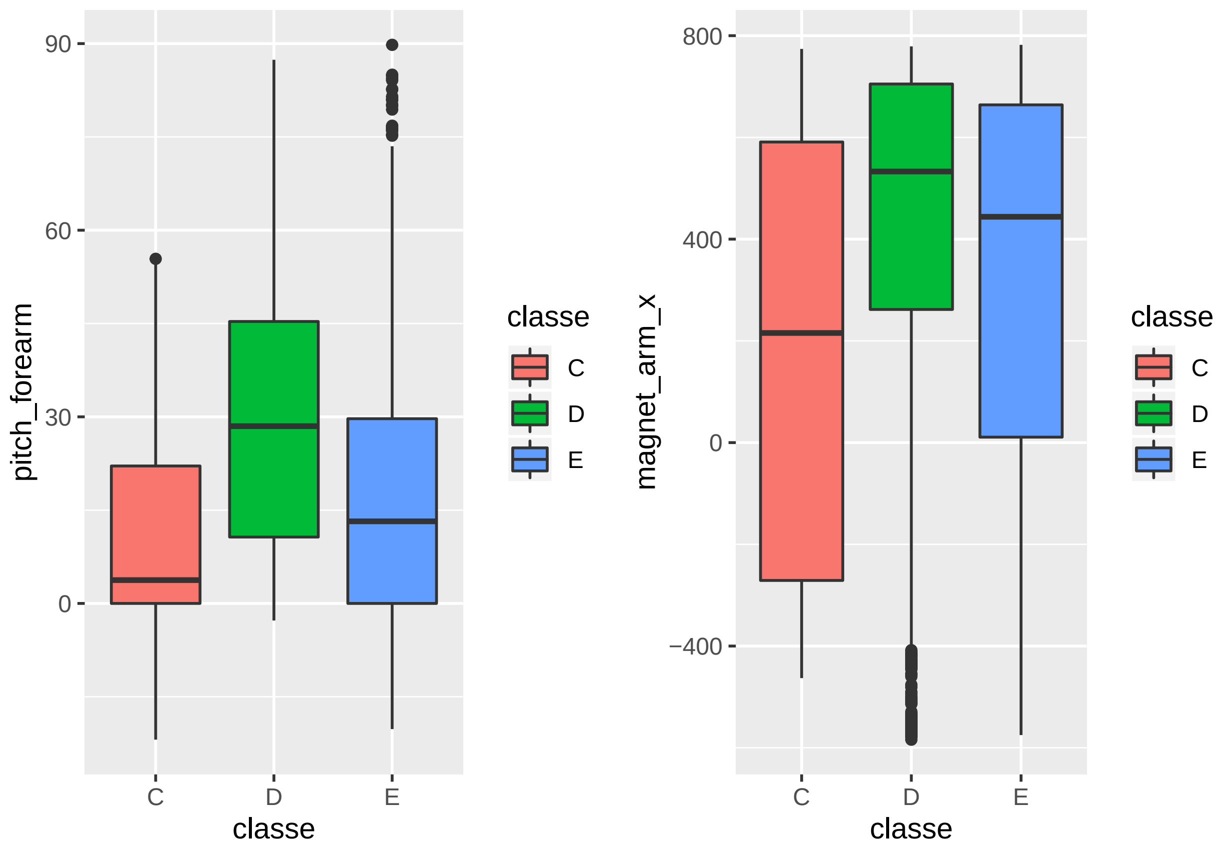
```
##             Var1 Var2       Freq
## 15 magnet_belt_y    A -0.3269371
```

Plotting of the two possible linear predictors

```
library(Rmisc)
```

```
## Warning: package 'Rmisc' was built under R version 3.6.1
```

```
p1 <- ggplot(training.subSetTrain, aes(classe,pitch_forearm)) +
  geom_boxplot(aes(fill=classe))
p2 <- ggplot(training.subSetTrain, aes(classe, magnet_arm_x)) +
  geom_boxplot(aes(fill=classe))
multiplot(p1,p2,cols=2)
```



This shows there is any wide separation among these correlated features

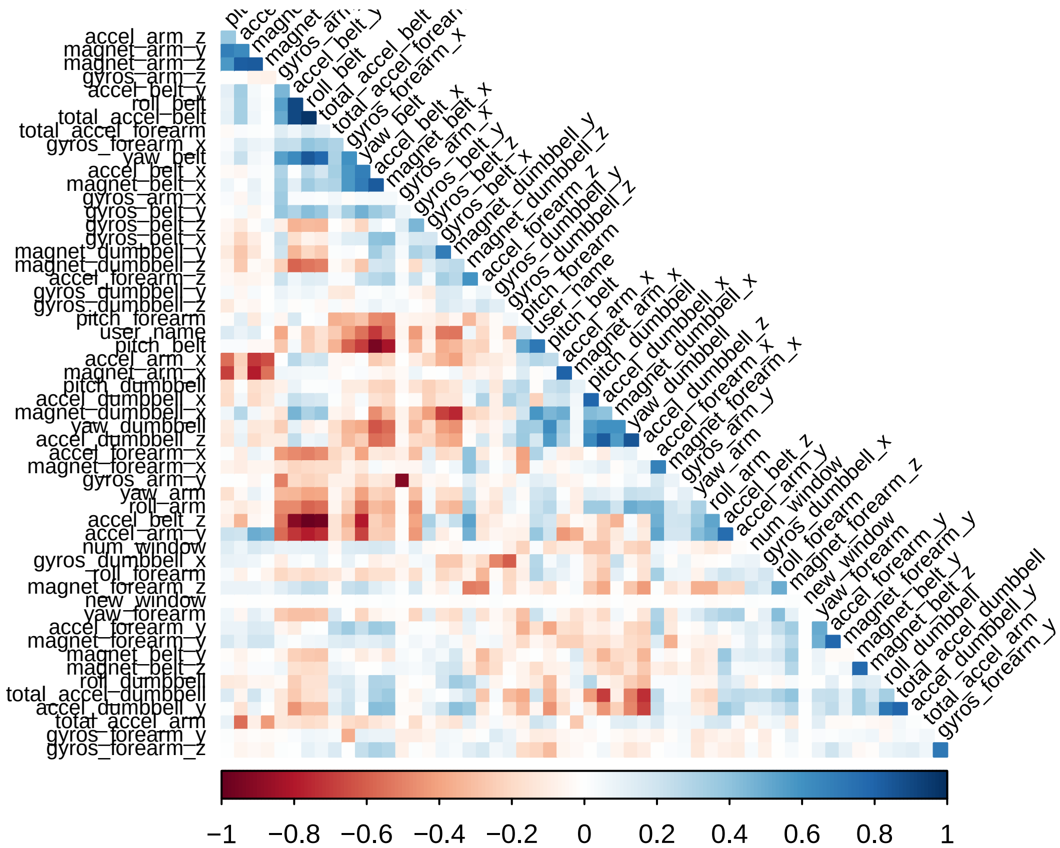
Model selection

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.6.1
```

```
## corrplot 0.84 loaded
```

```r
correlationMatrix <- cor(training.subSetTrain[, -classeIndex])
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.9, exact=TRUE)
excludeColumns <- c(highlyCorrelated, classeIndex)
corrplot(correlationMatrix, method="color", type="lower", order="hclust", tl.cex=0.70, tl.col="black",
```



This shows there are few correlated features. The model is to exclude them from PCA

```
pcaPreProcess.all <- preProcess(training.subSetTrain[, -classeIndex], method = "pca", thresh = 0.99)
training.subSetTrain.pca.all <- predict(pcaPreProcess.all, training.subSetTrain[, -classeIndex])
training.subSetTest.pca.all <- predict(pcaPreProcess.all, training.subSetTest[, -classeIndex])
testing.pca.all <- predict(pcaPreProcess.all, testing.cleaned[, -classeIndex])
pcaPreProcess.subset <- preProcess(training.subSetTrain[, -excludeColumns], method = "pca", thresh = 0.9
training.subSetTrain.pca.subset <- predict(pcaPreProcess.subset, training.subSetTrain[, -excludeColumns
training.subSetTest.pca.subset <- predict(pcaPreProcess.subset, training.subSetTest[, -excludeColumns])
testing.pca.subset <- predict(pcaPreProcess.subset, testing.cleaned[, -classeIndex])
```

Do the Random Forest training with 52 trees

```
memory.limit(size=500)
```

```
## Warning in memory.limit(size = 500): cannot decrease memory limit: ignored
```

```
## [1] 3973
```

```
ntree <- 52
start <- proc.time()
rfMod.cleaned <- randomForest(
  x=training.subSetTrain[, -classeIndex],
  y=training.subSetTrain$classe,
  xtest=training.subSetTest[, -classeIndex],
```

```r
  ytest=training.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE) #do.trace=TRUE
proc.time() - start
```

```
##    user  system elapsed
##   11.07    1.17   12.33
```

```r
start <- proc.time()
rfMod.exclude <- randomForest(
  x=training.subSetTrain[, -excludeColumns],
  y=training.subSetTrain$classe,
  xtest=training.subSetTest[, -excludeColumns],
  ytest=training.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE) #do.trace=TRUE
proc.time() - start
```

```
##    user  system elapsed
##    9.50    1.70   13.06
```

```r
start <- proc.time()
rfMod.pca.all <- randomForest(
  x=training.subSetTrain.pca.all,
  y=training.subSetTrain$classe,
  xtest=training.subSetTest.pca.all,
  ytest=training.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE) #do.trace=TRUE
proc.time() - start
```

```
##    user  system elapsed
##    8.82    1.68   28.72
```

```r
start <- proc.time()
rfMod.pca.subset <- randomForest(
  x=training.subSetTrain.pca.subset,
  y=training.subSetTrain$classe,
  xtest=training.subSetTest.pca.subset,
  ytest=training.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE) #do.trace=TRUE
proc.time() - start
```

```
##    user  system elapsed
##    8.78    2.98   90.81
```

Model examination

With the above fourtrained models, checking for the accuracies of each.

```r
rfMod.cleaned
```

```
##
## Call:
##  randomForest(x = training.subSetTrain[, -classeIndex], y = training.subSetTrain$classe,      xtest =
##                Type of random forest: classification
##                      Number of trees: 52
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.35%
## Confusion matrix:
##      C    D    E class.error
## C 2097    3    0 0.001428571
## D   14 2396    2 0.006633499
## E    0    6 2700 0.002217295
##                 Test set error rate: 0.29%
## Confusion matrix:
##     C   D   E class.error
## C 700   0   0 0.000000000
## D   4 799   1 0.006218905
## E   0   2 899 0.002219756
```

```r
rfMod.cleaned.training.acc <- round(1-sum(rfMod.cleaned$confusion[, 'class.error']),3)
paste0("Accuracy on training: ",rfMod.cleaned.training.acc)
```

```
## [1] "Accuracy on training: 0.99"
```

```r
rfMod.cleaned.testing.acc <- round(1-sum(rfMod.cleaned$test$confusion[, 'class.error']),3)
paste0("Accuracy on testing: ",rfMod.cleaned.testing.acc)
```

```
## [1] "Accuracy on testing: 0.992"
```

```r
rfMod.exclude
```

```
##
## Call:
##  randomForest(x = training.subSetTrain[, -excludeColumns], y = training.subSetTrain$classe,      xtes
##                Type of random forest: classification
##                      Number of trees: 52
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.39%
## Confusion matrix:
##      C    D    E class.error
## C 2098    2    0 0.000952381
## D   17 2392    3 0.008291874
## E    1    5 2700 0.002217295
```

```
##                  Test set error rate: 0.29%
## Confusion matrix:
##     C   D   E class.error
## C 700   0   0 0.000000000
## D   4 799   1 0.006218905
## E   0   2 899 0.002219756
```

```r
rfMod.exclude.training.acc <- round(1-sum(rfMod.exclude$confusion[, 'class.error']),3)
paste0("Accuracy on training: ",rfMod.exclude.training.acc)
```

```
## [1] "Accuracy on training: 0.989"
```

```r
rfMod.exclude.testing.acc <- round(1-sum(rfMod.exclude$test$confusion[, 'class.error']),3)
paste0("Accuracy on testing: ",rfMod.exclude.testing.acc)
```

```
## [1] "Accuracy on testing: 0.992"
```

```r
rfMod.pca.all
```

```
##
## Call:
##  randomForest(x = training.subSetTrain.pca.all, y = training.subSetTrain$classe,      xtest = traini
##                Type of random forest: classification
##                      Number of trees: 52
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 2.8%
## Confusion matrix:
##      C    D    E class.error
## C 2043   37   20  0.02714286
## D  101 2299   12  0.04684909
## E   16   16 2674  0.01182557
##                  Test set error rate: 2.2%
## Confusion matrix:
##     C   D   E class.error
## C 692   4   4 0.011428571
## D  37 763   4 0.050995025
## E   2   2 897 0.004439512
```

```r
rfMod.pca.all.training.acc <- round(1-sum(rfMod.pca.all$confusion[, 'class.error']),3)
paste0("Accuracy on training: ",rfMod.pca.all.training.acc)
```

```
## [1] "Accuracy on training: 0.914"
```

```r
rfMod.pca.all.testing.acc <- round(1-sum(rfMod.pca.all$test$confusion[, 'class.error']),3)
paste0("Accuracy on testing: ",rfMod.pca.all.testing.acc)
```

```
## [1] "Accuracy on testing: 0.933"
```

```r
rfMod.pca.subset
```

```
##
## Call:
##  randomForest(x = training.subSetTrain.pca.subset, y = training.subSetTrain$classe,      xtest = trai
##                Type of random forest: classification
##                      Number of trees: 52
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 3.08%
## Confusion matrix:
##      C    D    E class.error
## C 2044   34   22  0.02666667
## D   93 2298   21  0.04726368
## E   26   26 2654  0.01921656
##                 Test set error rate: 2.33%
## Confusion matrix:
##     C   D   E class.error
## C 692   4   4 0.011428571
## D  36 763   5 0.050995025
## E   3   4 894 0.007769145
```

```r
rfMod.pca.subset.training.acc <- round(1-sum(rfMod.pca.subset$confusion[, 'class.error']),3)
paste0("Accuracy on training: ",rfMod.pca.subset.training.acc)
```

```
## [1] "Accuracy on training: 0.907"
```

```r
rfMod.pca.subset.testing.acc <- round(1-sum(rfMod.pca.subset$test$confusion[, 'class.error']),3)
paste0("Accuracy on testing: ",rfMod.pca.subset.testing.acc)
```

```
## [1] "Accuracy on testing: 0.93"
```

Conclusion

This shows that PCA doesn't have a positive of the accuracy The `rfMod.exclude` perform's slightly better then the 'rfMod.cleaned'

Thus `rfMod.exclude` model is the best model to use for predicting the test set. It has an accuracy of 98.7% and an estimated out of bound error rate of 0.23%.

Test results

The `rfMod.exclude` is choosen as the best model. It is compared with other 3 models.

```r
predictions <- t(cbind(
    exclude=as.data.frame(predict(rfMod.exclude, testing.cleaned[, -excludeColumns]), optional=TRUE),
    cleaned=as.data.frame(predict(rfMod.cleaned, testing.cleaned), optional=TRUE),
    pcaAll=as.data.frame(predict(rfMod.pca.all, testing.pca.all), optional=TRUE),
    pcaExclude=as.data.frame(predict(rfMod.pca.subset, testing.pca.subset), optional=TRUE)
))
predictions
```

```
##            1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16
## exclude    "E" "C" "C" "E" "C" "E" "D" "C" "D" "D" "D" "C" "E" "C" "E" "E"
## cleaned    "D" "C" "C" "C" "C" "E" "D" "C" "D" "D" "C" "C" "E" "C" "E" "E"
## pcaAll     "D" "C" "C" "E" "C" "E" "D" "C" "E" "C" "D" "C" "E" "C" "E" "E"
## pcaExclude "C" "C" "C" "E" "C" "E" "D" "D" "E" "C" "C" "C" "E" "C" "E" "E"
##            17  18  19  20
## exclude    "E" "D" "D" "E"
## cleaned    "E" "D" "D" "E"
## pcaAll     "E" "E" "E" "E"
## pcaExclude "E" "E" "E" "E"
```