Prediction Assignment Writeup

Lesky Anatias

April 2, 2017

Summary

The goal of this project is to train a predictive model to predict the manner in which 6 participants did the exercise using data taken from accelerometers on the belt, forearm, arm, and dumbell based on a dataset provided by HAR link (http://groupware.les.inf.puc-rio.br/har).

The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The five ways are exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Only Class A corresponds to correct performance.

Pre-processing and Exploratory Analyses

```
training.url <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testing.url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training.file <- "./data/pml-training.csv"
testing.file  <- "./data/pml-testing.csv"
if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists(training.file)) {
  download.file(training.url, destFile=training.file, method="curl")
}
if (!file.exists(testing.file)) {
  download.file(testing.url, destFile=testing.file, method="curl")
}

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

dim(training.raw)
```

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

```
dim(testing.raw)
```

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

```
# Excluded from report:
# str(training.raw)
# summary(training.raw)
```

From the exploratory analyses, we found out the training dataset contains 19622 observations and 160 variables, and the testing dataset contains 20 observations and 160 variables. The “classe” variable in the training set is the outcome to predict.

Data Cleaning

Before working with the model prediction, we need to clean the data by eliminating observations with NA / empty values / other unrelated data.

```
sum(complete.cases(training.raw))
```

```
## [1] 406
```

```
#Remove columns that contain NA / empty values
training.raw <- training.raw[, colSums(is.na(training.raw)) == 0]
testing.raw <- testing.raw[, colSums(is.na(testing.raw)) == 0]
#Remove columns that do not contribute to the accelerometer measurements.
classe <- training.raw$classe
training.remove <- grepl("^X|timestamp|window", names(training.raw))
training.raw <- training.raw[, !training.remove]
training.cleaned <- training.raw[, sapply(training.raw, is.numeric)]
training.cleaned$classe <- classe
testing.remove <- grepl("^X|timestamp|window", names(testing.raw))
testing.raw <- testing.raw[, !testing.remove]
testing.cleaned <- testing.raw[, sapply(testing.raw, is.numeric)]
```

Now, the cleaned training data set contains 19622 observations and 53 variables, while the testing data set contains 20 observations and 53 variables. The “classe” variable is still in the cleaned training set.

Data Partition

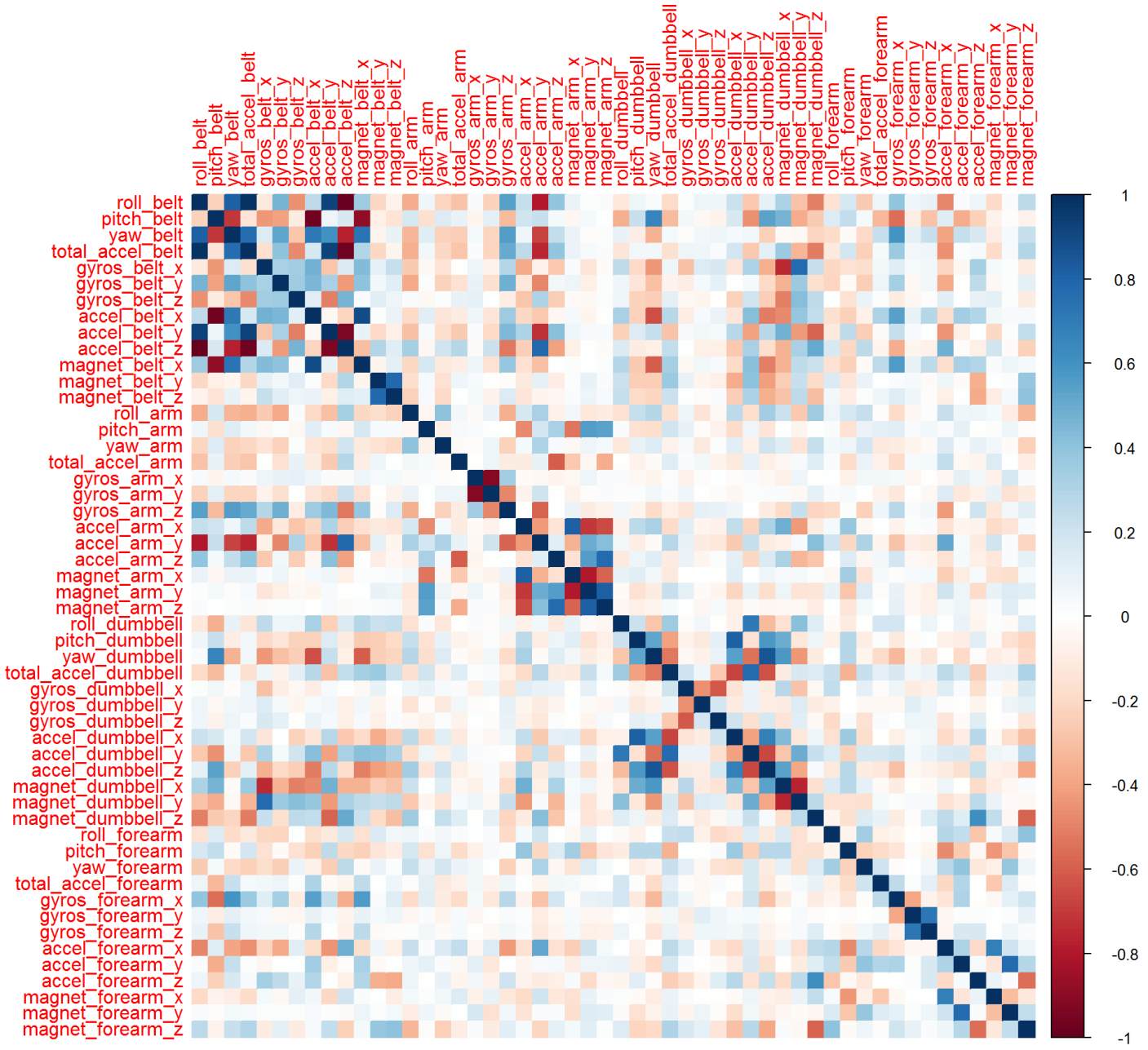
Since the test dataset is the ultimate validation dataset, we partition cleaned training dataset into a test dataset (75%) and a train dataset (25%). The train dataset will be used to conduct cross validation.

```
set.seed(210679)
pTrain <- createDataPartition(training.cleaned$classe, p=0.75, list=FALSE)
training.data <- training.cleaned[pTrain, ]
testing.data <- training.cleaned[-pTrain, ]
```

Data Modeling

Then, we identify variables with high correlations amongst each other in our dataset with correlation matrix below:

```
corr.plot <- cor(training.data[, -length(names(training.data))])
corrplot(corr.plot, method="color")
```



We see that there are some features that are quite correlated with each other.

Since *Classification Tree* method does not perform well (see appendix), We fit a predictive model for activity recognition using *Random Forest* algorithm because it automatically selects important variables and is robust to correlated covariates & outliers. We use *5-fold cross validation* and *250 trees*.

```
rf.control <- trainControl(method="cv", 5)
rf.model <- train(classe ~ ., data=training.data, method="rf", trControl=rf.control, ntree=250)
rf.model
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11773, 11774, 11775, 11776, 11774
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9925940  0.9906310
##   27    0.9917108  0.9895142
##   52    0.9863425  0.9827217
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Then, we estimate the performance of the model on the validation data set.

```
rf.prediction <- predict(rf.model, testing.data)
confusionMatrix(testing.data$classe, rf.prediction)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    1    0    0    0
##          B   14  933    2    0    0
##          C    0    8  847    0    0
##          D    0    0   13  791    0
##          E    0    0    0    1  900
##
## Overall Statistics
##
##                Accuracy : 0.992
##                  95% CI : (0.9891, 0.9943)
##     No Information Rate : 0.2871
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9899
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9901   0.9904   0.9826   0.9987   1.0000
## Specificity            0.9997   0.9960   0.9980   0.9968   0.9998
## Pos Pred Value         0.9993   0.9831   0.9906   0.9838   0.9989
## Neg Pred Value         0.9960   0.9977   0.9963   0.9998   1.0000
## Prevalence             0.2871   0.1921   0.1758   0.1615   0.1835
## Detection Rate         0.2843   0.1903   0.1727   0.1613   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9949   0.9932   0.9903   0.9978   0.9999
```

```
accuracy <- postResample(rf.prediction, testing.data$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9920473 0.9899373
```

```
error <- 1 - as.numeric(confusionMatrix(testing.data$classe, rf.prediction)$overall[1])
error
```

```
## [1] 0.007952692
```

So, the estimated accuracy of the model is 99.2% and the estimated out-of-sample error is 0.79%. This may be due to the fact that many predictors are highly correlated. Random forests chooses a subset of predictors at each split and decorrelate the trees. This leads to high accuracy.

Predicting the Test Dataset

Now, we use use the *random forests* to predict the outcome variable `classe` for the testing set. We remove the `problem_id` column first.

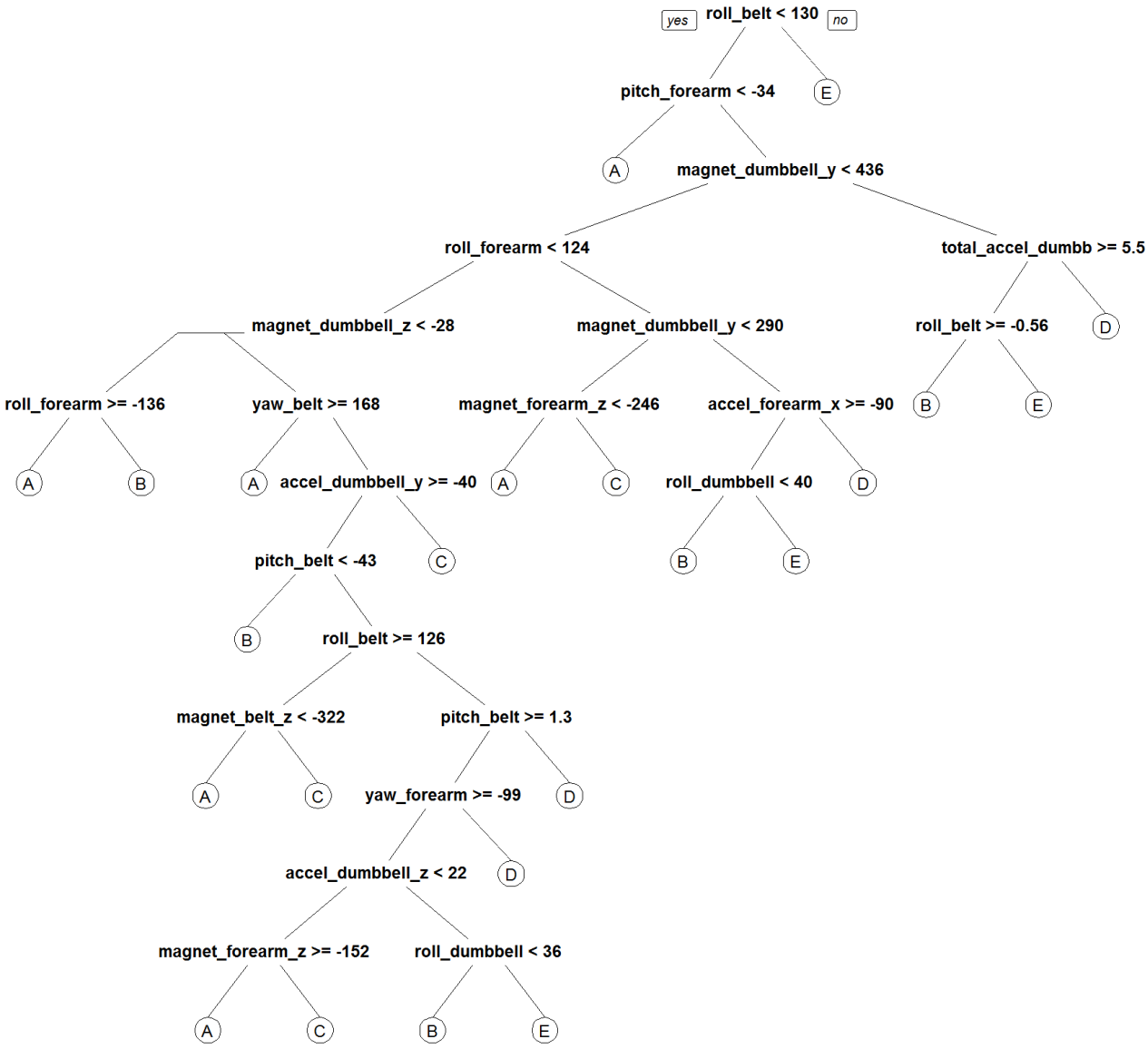
```
test.result <- predict(rf.model, testing.cleaned[, -length(names(testing.cleaned))])
test.result
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

Appendix

Classification Trees

```
tree.model <- rpart(classe ~ ., data=training.data, method="class")
prp(tree.model)
```




```
ct.control <- trainControl(method="cv", 5)
ct.model <- train(classe ~ ., data=training.data, method="rpart", trControl=ct.control)
ct.model
```

```
## CART
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11776, 11773, 11776, 11772, 11775
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03550745  0.5250872  0.38908347
##   0.06035001  0.4164232  0.20914089
##   0.11392766  0.3314305  0.07176123
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03550745.
```

```
ct.prediction <- predict(ct.model, testing.data)
confusionMatrix(testing.data$classe, ct.prediction)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1265   29   99    0    2
##          B  387  326  236    0    0
##          C  400   28  427    0    0
##          D  352  164  288    0    0
##          E  125  116  241    0  419
##
## Overall Statistics
##
##                Accuracy : 0.4969
##                  95% CI : (0.4829, 0.511)
##     No Information Rate : 0.5157
##     P-Value [Acc > NIR] : 0.9959
##
##                   Kappa : 0.3428
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5002  0.49170  0.33075       NA  0.99525
## Specificity            0.9453  0.85310  0.88154   0.8361  0.89248
## Pos Pred Value         0.9068  0.34352  0.49942       NA  0.46504
## Neg Pred Value         0.6398  0.91479  0.78661       NA  0.99950
## Prevalence             0.5157  0.13520  0.26325   0.0000  0.08585
## Detection Rate         0.2580  0.06648  0.08707   0.0000  0.08544
## Detection Prevalence   0.2845  0.19352  0.17435   0.1639  0.18373
## Balanced Accuracy      0.7227  0.67240  0.60615       NA  0.94387
```

```
ct.accuracy <- postResample(ct.prediction, testing.data$classe)
ct.accuracy
```

```
##  Accuracy     Kappa
## 0.4969413 0.3428105
```

```
ct.error <- 1 - as.numeric(confusionMatrix(testing.data$classe, ct.prediction)$overall[1])
ct.error
```

```
## [1] 0.5030587
```

From the confusion matrix, the estimated accuracy of the model is 50% and the estimated out-of-sample error is 50%. Using classification tree does not predict the outcome `classe` very well.