Practical Machine Learning Project

Executive Summary

One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The goal of the project is to predict the manner in which a participant did the exercise. This is the "classe" variable in the training set.

Data Review

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: 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).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

Analysis

1. The data sets will be read in from the url supplied.
2. Some exploratory data analysis will be done.
3. The data will be cleaned.
4. A few different models will be fitted.
5. The out-of-sample error will be predicted.
6. The best model will be used to predict how well an activity is done.

```
train_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

download.file(train_url, destfile = "training.csv")
download.file(test_url, destfile = "testing.csv")
```

Explore the data set information

```
# Set seed to allow reproducibility
set.seed(120021)
train <- read_csv("training.csv")
test <- read_csv("testing.csv")
# investigate training data
dim(train)
```

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

```
# head(train)
# str(train)
# summary(train)
```

Clean the data set

We see quite a number of NAs in the data exploration, as well as some division by 0's.

```
# Let's keep all columns that have 75% of actual data
seventy_five_perc <- 0.75 * nrow(train)
how_many_nas <- apply(train, 2, function(x){sum(is.na(x))})
col_to_keep <- which(how_many_nas < seventy_five_perc)
train_cleaned <- train[,col_to_keep]
# Do the same for the test data set
test_cleaned <- test[,col_to_keep]

# There are also some timestamp and window columns that we should remove
col_to_rem <- grep("timestamp|window|^X", names(train_cleaned))
train_cleaned <- train_cleaned[,-col_to_rem]
# Do the same for the test data set
test_cleaned <- test_cleaned[,-col_to_rem]

# Make the classe variable a factor
train_cleaned$classe <- factor(train_cleaned$classe)
```

Create the training and testing data sets

```
inTrain <- createDataPartition(y=train_cleaned$classe, p=0.75, list=F)
train_set <- train_cleaned[inTrain, ]
train_test_set <- train_cleaned[-inTrain, ]
dim(train_set)
```

```
## [1] 14718    54
```

```
dim(train_test_set)
```

```
## [1] 4904   54
```

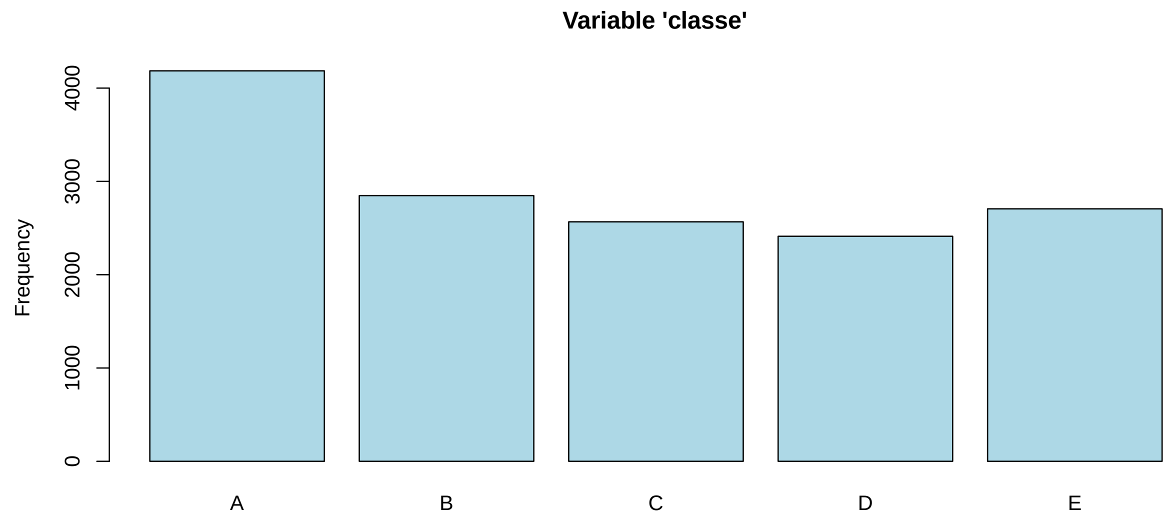
```
train_set = na.omit(train_set)
train_test_set = na.omit(train_test_set)
dim(train_set)
```

```
## [1] 14718    54
```

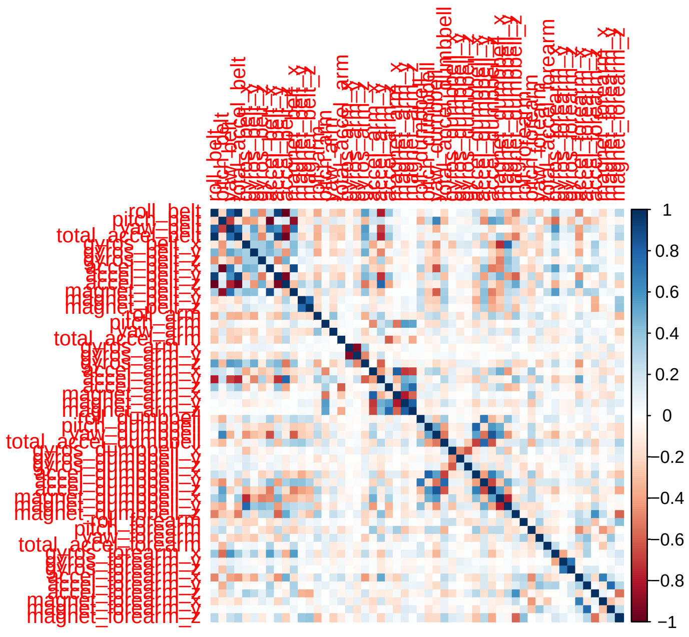
```
dim(train_test_set)
```

```
## [1] 4903   54
```

```
# What does the classe levels look like
plot(train_set$classe, col="light blue", main="Variable 'classe'", xlab="", ylab="Frequency")
```



```
# What does the Correlation Matrix look like
classe_col_num <- which(names(train_set) == "classe")
correlationMatrix <- cor(train_set[, -c(1,classe_col_num)],use="pairwise.complete.obs")
corrplot(correlationMatrix, method="color")
```



Training Models

```
# Fit models and check differences

# Decision Tree
modelFit1 <- rpart(classe~., data=train_set,  method="class")
predict1 <- predict(modelFit1, newdata=train_test_set, type="class")
```

```r
cf1 <- confusionMatrix(predict1, train_test_set$classe)
cf1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1245  133   46   93   27
##          B   56  554   42   66   69
##          C   41   83  661  125  120
##          D   28   72   56  434   41
##          E   24  107   50   86  644
##
## Overall Statistics
##
##                Accuracy : 0.7216
##                  95% CI : (0.7088, 0.7341)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6465
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8931   0.5838   0.7731  0.53980   0.7148
## Specificity            0.9148   0.9411   0.9088  0.95194   0.9333
## Pos Pred Value         0.8063   0.7039   0.6417  0.68780   0.7069
## Neg Pred Value         0.9556   0.9040   0.9499  0.91339   0.9356
## Prevalence             0.2843   0.1936   0.1744  0.16398   0.1838
## Detection Rate         0.2539   0.1130   0.1348  0.08852   0.1313
## Detection Prevalence   0.3149   0.1605   0.2101  0.12870   0.1858
## Balanced Accuracy      0.9040   0.7624   0.8410  0.74587   0.8240
```

```r
# Random Forest
fitControl <- trainControl(method="cv", number=5, verboseIter = F, preProcOptions = "pca")
modelFit2 <- train(classe~., data=train_set, method="rf", trControl=fitControl, verbose=F)
predict2 <- predict(modelFit2, newdata=train_test_set)
confusionMatrix(predict2, train_test_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1390    8    0    0    0
##          B    3  939    4    0    1
##          C    1    2  849    6    2
##          D    0    0    2  798    0
##          E    0    0    0    0  898
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9915, 0.996)
##     No Information Rate : 0.2843
##     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            0.9971   0.9895   0.9930   0.9925   0.9967
## Specificity            0.9977   0.9980   0.9973   0.9995   1.0000
## Pos Pred Value         0.9943   0.9916   0.9872   0.9975   1.0000
## Neg Pred Value         0.9989   0.9975   0.9985   0.9985   0.9993
## Prevalence             0.2843   0.1936   0.1744   0.1640   0.1838
## Detection Rate         0.2835   0.1915   0.1732   0.1628   0.1832
## Detection Prevalence   0.2851   0.1931   0.1754   0.1632   0.1832
## Balanced Accuracy      0.9974   0.9937   0.9951   0.9960   0.9983
```

```
# Support Vector Machine - linear
modelFit3 <- train(classe~., data=train_set, method="svmLinear", trControl=fitControl)
predict3 <- predict(modelFit3, newdata=train_test_set)
#nrow(train_test_set); length(predict3)
confusionMatrix(predict3, train_test_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1291  121   61   56   29
##          B   24  702   85   29   97
##          C   34   48  684   93   53
##          D   44   10   13  603   47
##          E    1   68   12   23  675
##
## Overall Statistics
##
##                Accuracy : 0.8066
##                  95% CI : (0.7953, 0.8176)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7543
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9261   0.7397   0.8000   0.7500   0.7492
## Specificity            0.9239   0.9406   0.9437   0.9722   0.9740
## Pos Pred Value         0.8286   0.7492   0.7500   0.8410   0.8665
## Neg Pred Value         0.9692   0.9377   0.9572   0.9520   0.9452
## Prevalence             0.2843   0.1936   0.1744   0.1640   0.1838
## Detection Rate         0.2633   0.1432   0.1395   0.1230   0.1377
## Detection Prevalence   0.3178   0.1911   0.1860   0.1462   0.1589
## Balanced Accuracy      0.9250   0.8401   0.8718   0.8611   0.8616
```

```r
# LogitBoost
modelFit4 <- train(classe~., data=train_set, method="LogitBoost", trControl=fitControl)
predict4 <- predict(modelFit4, newdata=train_test_set)
confusionMatrix(predict4, train_test_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1298   89   25   13    8
##          B   11  588   28    2   22
##          C    6   62  636   65   26
##          D   10   12   10  552   14
##          E    3   23    1   10  735
##
## Overall Statistics
##
##                Accuracy : 0.8964
##                  95% CI : (0.8869, 0.9055)
##     No Information Rate : 0.3125
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8673
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9774   0.7597   0.9086   0.8598   0.9130
## Specificity            0.9538   0.9819   0.9552   0.9872   0.9893
## Pos Pred Value         0.9058   0.9032   0.8000   0.9231   0.9521
## Neg Pred Value         0.9893   0.9483   0.9815   0.9753   0.9799
## Prevalence             0.3125   0.1822   0.1647   0.1511   0.1895
## Detection Rate         0.3055   0.1384   0.1497   0.1299   0.1730
## Detection Prevalence   0.3373   0.1532   0.1871   0.1407   0.1817
## Balanced Accuracy      0.9656   0.8708   0.9319   0.9235   0.9512
```

```r
models <- c("Decision Tree", "Random Forest", "Support Vector Machine - linear", "LogitBoost")
accuracy <- c(max(cf1$overall[1]), max(modelFit2$results$Accuracy),
              max(modelFit3$results$Accuracy), max(modelFit4$results$Accuracy))
results <- cbind(models, accuracy)
results
```

```
##      models                            accuracy
## [1,] "Decision Tree"                   "0.721599021007546"
## [2,] "Random Forest"                   "0.990896113884851"
## [3,] "Support Vector Machine - linear" "0.801873647656948"
## [4,] "LogitBoost"                      "0.895969890767951"
```

Results

The Random Forest performs best from the models fitted. This will be used to predict the quality of activity done by a participant. It must be kept in mind that the data was collected utilising healthy particpants. The model used for prediction may not do as well for a different population.