```
library(knitr)
opts_chunk$set(cache = TRUE)
```

Predictive Model of Quality of Weight-Lifting Exercises

Synopsis

A Random Forest model is trained to recognise different variations in quality of weight-lifting acvitity. The model achieves an estimated out-of-sample accuracy of 99%.

I. Data Processing

First, we download a dataset containing sensor measurement output from a number of volunteers performing five variations of a barbell weight lift.

```
# Download data and load 'caret' package for modeling
setInternet2(TRUE)
dat <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv')
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

The downloaded dataset contains a total of **19622** labeled observations.

To clean-up this data set for analysis and modeling, we remove columns that: * Are irrelevant features, such as timestamps and subject names * Contain mostly "NA" values * Are predominantly blank

```
# Remove unneccessary non-feature columns
nonFeatureColNames <-
      c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2',
      'cvtd_timestamp', 'new_window', 'num_window')
dat <- dat[, -match(nonFeatureColNames, names(dat))]
rm(nonFeatureColNames)

# Remove columns with over 30% of values being NAs
naPropUnacceptable <- 0.3
naProp <- colSums(is.na(dat)) / nrow(dat)
naColNums <- (1:ncol(dat))[naProp > naPropUnacceptable]
dat <- dat[, -naColNums]
rm(naPropUnacceptable, naProp, naColNums)

# Remove columns with many blanks (read as "factor" class)
colClasses <- sapply(dat, class)
factorColNums <- (1:ncol(dat))[colClasses == 'factor']
factorColNums <- factorColNums[1 : (length(factorColNums) - 1)]
dat <- dat[, -factorColNums]
rm(colClasses, factorColNums)
numFeats <- ncol(dat) - 1
```

The number of features used for modelling is **52**.

Lastly, we randomly split the dataset into a Training set (60%) and a Testing set (40%).

```
# Split data set into Training & Testing set
set.seed(313)
indcsTrain <- createDataPartition(y = dat$classe, p = 0.6, list = FALSE)
datTrain <- dat[indcsTrain, ]
datTest <- dat[-indcsTrain, ]
rm(dat, indcsTrain)
```

The Training set contains **11776** observations, and the Test set contains **7846**.

II. Training a Random Forest Classification Model

A Random Forest model is fit to the Training set to predict the activity quality variable **classe** on the 52 selected features. The training involves **4-fold cross-validation**.

```
# Train a Random Forest model
model <- train(classe ~ ., data = datTrain, method = 'rf',
      trControl = trainControl(method = "cv", number = 4),
      allowParallel = TRUE)
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

III. Estimated Out-of-Sample Accuracy

The trained model achieves high predictive accuracy on the seperate Testing dataset, with Balanced Accuracy of **over 99%** for each of the five activity quality variations A-E.

```
predTest <- predict(model, datTest)
confusionMatrix(predTest, datTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    8    0    0    0
##          B    1 1509   18    0    0
##          C    0    1 1349   23    7
##          D    0    0    1 1262    0
##          E    0    0    0    1 1435
##
## Overall Statistics
##
##                Accuracy : 0.9924
##                  95% CI : (0.9902, 0.9942)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9903
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9941   0.9861   0.9813   0.9951
## Specificity            0.9986   0.9970   0.9952   0.9998   0.9998
## Pos Pred Value         0.9964   0.9876   0.9775   0.9992   0.9993
## Neg Pred Value         0.9998   0.9986   0.9971   0.9964   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1923   0.1719   0.1608   0.1829
## Detection Prevalence   0.2854   0.1947   0.1759   0.1610   0.1830
## Balanced Accuracy      0.9991   0.9955   0.9907   0.9906   0.9975
```

Thus, we expect an **out-of-sample error rate of only approximately 1%**.

Finally, the model also performs well in the *Practical Machine Learning* course's separate testing dataset, classifying correctly 20 out of 20 examples.