(Last updated: 2019-12-26)

This is a course project for 'Practical Machine Learning'. It aims to predict the type/class of exercise in the test set, based on a model developed using the training set data. This report has the following sections:

1. Data access and processing

2. Model development, including justification, cross validation, and expected out of sample error.
3. Results of prediction model for 20 test cases

Note to the peer graders: I had lot of problems and questions for this assignment. I would love to learn from your experience/code, and would very much appreciate specific suggestions to improve this assignment. Thank you in advance!

1. DATA PROCESSING

1.1 Overall data structure

First access the train and test data sets.

```
#Get data
#train <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
#test <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
setwd("C:/Users/YoonJoung Choi/Dropbox/2 R/2 R Coursera/8PracticalMachineLearning")
train <- read.csv("pml-training.csv")
test <- read.csv("pml-testing.csv")

#Check data
nobstest<-nrow(test)
nobstrain<-nrow(train)
nvar<-ncol(train)
obspeople<-length(unique(train$user_name))
```

There are 20 observations in the test data, and 19622 in the train data set. The train data set has 160 variables, including the outcome (i.e., **classe**, movement type), measured among 6 study participants.

1.2 Outcome

In **classe**, there are five different exercise types (A, B, C, D, and E). Class A is the correct movement, while the rest four classes are mistakes. They 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)
- throwing the hips to the front (Class E)

Below is the distribution of the exercise pattern in the train data set (n=19622).

```
   A    B    C    D    E 
5580 3797 3422 3216 3607 
```

1.3 Covariates and data management

There are so many potential covariates. Let's check them out briefly.

- Variables with lot of missing values?

```
#check train data set first
nNA<-sapply(train, function(x) sum(is.na(x)))
var<-as.character(names(train))
tableTrain<-as.data.frame(cbind(var, nNA))
crosstab<-table(tableTrain$nNA)

table(tableTrain$nNA)
```

```

    0 19216 
   93    67 
```

In the training data, while 93 variables have no missing values, 67 variables have missing values in most rows (19216 out of 19622). We should drop those 67 variables from the analysis.

```
#check test data
nNA<-sapply(test, function(x) sum(is.na(x)))
var<-as.character(names(test))
tableTest<-as.data.frame(cbind(var, nNA))
crosstab<-table(tableTest$nNA)

table(tableTest$nNA)
```

```

  0  20 
 60 100 
```

In the test data, 100 variables have missing values in *all 20 rows*. We should drop all of these variables from the analysis.

Below shows that the list of variables missing in train data is a subset of the list in test data. Thus, drop the list based on the test data from both train and test datasets.

```
library(dplyr)

#define variables with missing values in train
dropTrain<-tableTrain %>%
    subset(nNA!=0) %>%
    select(var)
dropvarTrain<-as.vector(as.character(dropTrain$var))

#define variables with missing values in test
dropTest<-tableTest %>%
    subset(nNA!=0) %>%
    select(var)
dropvarTest<-as.vector(as.character(dropTest$var))
```

```
#define variables with missing values in BOTH
common <- intersect(dropvarTrain, dropvarTest)
length(dropvarTrain)
```

```
[1] 67
```

```
length(dropvarTest)
```

```
[1] 100
```

```
length(common)
```

```
[1] 67
```

```
#delete the list of variables from both train and test. Use these datasets for modeling
trainNEW<-select(train, -(dropvarTest))
testNEW<-select(test, -(dropvarTest))
```

- Variables with too many categories (i.e., factor variables that are actually numeric)?

```
var<-as.character(names(trainNEW))
class<-sapply(trainNEW, function(x) class(x) )
isfactor<-sapply(trainNEW, function(x) is.factor(x) )
nUnique<- sapply(trainNEW, function(x) length(unique(x)) )
tabletrain<-as.data.frame(cbind(var, class, isfactor, nUnique))

#keep only factor variables AND check the number of unique values in those factor variables
table<-subset(tabletrain, isfactor==TRUE)
table$nUnique <- as.numeric(table$nUnique)
table(table$nUnique)
```

```

21 22 43 46
 1  1  1  1
```

- Different variable class between between train and test data sets?

```
varTEST<-as.character(names(testNEW))
classTEST<-sapply(testNEW, function(x) class(x) )
compare <- as.data.frame(cbind(var, class, varTEST, classTEST) )
    for (i in 1:4) {
        compare[ , i]<-as.character(compare[ , i])
    }
table(compare$class, compare$classTEST)
```

```

          factor integer numeric
  factor       3       1       0
  integer      0      29       0
  numeric      0       3      24
```

```r
nmismatch<-nrow(subset(compare, compare$class!=compare$classTEST))
```

There are 4 variables that are in different classes between the two data sets. This will be taken care of later right before prediction.

- Finally, ensure same levels for factor variables between train and test data sets.

```r
common <- intersect(names(trainNEW), names(testNEW))

for (p in common) {
  if (class(trainNEW[[p]]) == "factor") {
    levels(testNEW[[p]]) <- levels(trainNEW[[p]])
  }
}
```

2. MODEL DEVELOPMENT

2.1 Model selection

There are a few things to consider to choose a modeling approach.
- The outcome is a categorical variable, thus a non-linear function is need.
- There are over 90 covariates, thus methods that can handle a large number of covariates would be helpful (unless we summarize/reduce covariates by using PCA, for example. *(But, a quetion: is it correct that PCA is used for linear models mainly?)*
- Observations are potentially correlated within each person. If that's the case and modeling does not address that, estimates may be biased. *(But, another question: I really don't know how to handle that issue. would apprecaite any suggestions on this)*

I chose to use Random forest, partially because of its built-in cross validation function, since it basically combines multiple decision trees (from bagging) to determine the final output.

Before moving on, by the way, there are so many observations (19622!), and it takes way too long to run even one model. So, let's randomly select 20% of train data and use that for model fitting train data **(trainNEWsub)**. The rest will be used for cross-validation **(trainNEWvalidation)**.

```r
set.seed(1234)
library(caret)

inSubsample <- createDataPartition(y=trainNEW$classe,
                              p=0.20, list=FALSE)
trainNEWsub <- trainNEW[inSubsample,]
trainNEWvalidation <- trainNEW[-inSubsample,]
```

Now, let's fit the first model using default parameters.

```r
set.seed(1234)

# default train control
trControl <- trainControl(method = "cv",
    number = 3,
    search = "grid")
```

```r
# Run the model
rf_default <- train(classe~.,
    data = trainNEWsub,
    method = "rf",
    metric = "Accuracy",
    trControl = trControl,
    importance = TRUE)
# Print the results
print(rf_default)
```

```
Random Forest

3927 samples
  59 predictor
   5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 2617, 2618, 2619
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.9806495  0.9755164
  41    0.9997452  0.9996777
  81    0.9994907  0.9993559

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 41.
```

Then, refine the model by changing the number of trees and the number of covariates (i.e., the number of variables randomly sampled at each stage).

```r
set.seed(1234)

# set mtry to try
tuneGrid <- expand.grid(.mtry = c(7, 14, 21, 28))
# three different ntree: 50, 100, 200
model1 <- train(classe~., trainNEWsub, method = "rf", metric = "Accuracy",
                trControl = trControl, importance = TRUE, tuneGrid = tuneGrid,
                ntree = 50)
print(model1)
```

```
Random Forest

3927 samples
  59 predictor
   5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 2617, 2618, 2619
Resampling results across tuning parameters:
  mtry  Accuracy   Kappa
   7    0.9974545  0.9967806
  14    1.0000000  1.0000000
  21    1.0000000  1.0000000
  28    1.0000000  1.0000000

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 14.
```

```
model2 <- train(classe~., trainNEWsub, method = "rf", metric = "Accuracy",
                trControl = trControl, importance = TRUE, tuneGrid = tuneGrid,
                ntree = 100)
print(model2)
```

```
Random Forest

3927 samples
  59 predictor
   5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 2618, 2618, 2618
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   7    0.9964349  0.9954908
  14    0.9997454  0.9996779
  21    0.9997454  0.9996779
  28    0.9997454  0.9996779

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 14.
```

```
model3 <- train(classe~., trainNEWsub, method = "rf", metric = "Accuracy",
                trControl = trControl, importance = TRUE, tuneGrid = tuneGrid,
                ntree = 200)
print(model3)
```

```
Random Forest

3927 samples
  59 predictor
   5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 2616, 2618, 2620
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   7    0.9956671  0.9945189
  14    0.9987252  0.9983875
  21    0.9994899  0.9993548
  28    0.9997457  0.9996785

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 28.
```

2.2 Assess accuracy and select a final model

```r
pred1 <-predict(model1, trainNEWvalidation)
pred2 <-predict(model2, trainNEWvalidation)
pred3 <-predict(model3, trainNEWvalidation)

confusionMatrix(pred1, trainNEWvalidation$classe)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 4464    4    0    0    0
         B    0 3033    0    0    0
         C    0    0 2737    0    0
         D    0    0    0 2572    0
         E    0    0    0    0 2885

Overall Statistics

               Accuracy : 0.9997
                 95% CI : (0.9993, 0.9999)
    No Information Rate : 0.2844
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9997
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            1.0000   0.9987   1.0000   1.0000   1.0000
Specificity            0.9996   1.0000   1.0000   1.0000   1.0000
Pos Pred Value         0.9991   1.0000   1.0000   1.0000   1.0000
Neg Pred Value         1.0000   0.9997   1.0000   1.0000   1.0000
Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
Detection Rate         0.2844   0.1932   0.1744   0.1639   0.1838
Detection Prevalence   0.2847   0.1932   0.1744   0.1639   0.1838
Balanced Accuracy      0.9998   0.9993   1.0000   1.0000   1.0000
```

```r
confusionMatrix(pred2, trainNEWvalidation$classe)
```

```
Confusion Matrix and Statistics

          Reference
```

```
Prediction    A    B    C    D    E
         A 4464    2    0    0    0
         B    0 3035    0    0    0
         C    0    0 2737    0    0
         D    0    0    0 2572    0
         E    0    0    0    0 2885

Overall Statistics

               Accuracy : 0.9999
                 95% CI : (0.9995, 1)
    No Information Rate : 0.2844
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9998
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            1.0000   0.9993   1.0000   1.0000   1.0000
Specificity            0.9998   1.0000   1.0000   1.0000   1.0000
Pos Pred Value         0.9996   1.0000   1.0000   1.0000   1.0000
Neg Pred Value         1.0000   0.9998   1.0000   1.0000   1.0000
Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
Detection Rate         0.2844   0.1934   0.1744   0.1639   0.1838
Detection Prevalence   0.2845   0.1934   0.1744   0.1639   0.1838
Balanced Accuracy      0.9999   0.9997   1.0000   1.0000   1.0000
```

```r
confusionMatrix(pred3, trainNEWvalidation$classe)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 4464    2    0    0    0
         B    0 3035    0    0    0
         C    0    0 2737    0    0
         D    0    0    0 2572    0
         E    0    0    0    0 2885

Overall Statistics

               Accuracy : 0.9999
                 95% CI : (0.9995, 1)
    No Information Rate : 0.2844
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9998
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
```

```
Sensitivity            1.0000   0.9993   1.0000   1.0000   1.0000
Specificity            0.9998   1.0000   1.0000   1.0000   1.0000
Pos Pred Value         0.9996   1.0000   1.0000   1.0000   1.0000
Neg Pred Value         1.0000   0.9998   1.0000   1.0000   1.0000
Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
Detection Rate         0.2844   0.1934   0.1744   0.1639   0.1838
Detection Prevalence   0.2845   0.1934   0.1744   0.1639   0.1838
Balanced Accuracy      0.9999   0.9997   1.0000   1.0000   1.0000
```

Above confusion matrices, based on evaluation of model performance on validation data, suggest that model 2 and 3 may be most appropriate. Between the two, I chose one with a less number of trees. **But, I'm worried about over fitting or some other problems - how can it have near 100% accuracy with the validation data?!** Anyhow, move along…

3. PREDICTION RESULTS

Finally, prediction!

```r
#predict
predtest2 <- predict(model2, testNEW)
predtest2
```

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

(Final note: well, is this complete failure or, in fact, do all test cases have some outcome? This is best I can do, and I'm not sure…)

REFERENCE

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz681eWk8h6