PRACTICAL MACHINE LEARNING COURSE PROJECT

Background Information

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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, we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data Source

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project comes from this original source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Loading Libraries

```
library(caret)
library(rpart)
library(randomForest)
```

Obtaining Data

We begin by reading both the csv files into two data frames. The process below would only involve storing the data in memory and not on disk.

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

pml_train <- read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
                      , na.strings=c("NA","#DIV/0!",""))
pml_test <- read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
                     , na.strings=c("NA","#DIV/0!",""))

dim(pml_train)
dim(pml_test)
```

We see that the training set contains 19622 observations and 160 variables while the testing set has 20 observations and 160 variables.

Cleaning the Data

We the proceed to first remove all the columns with missing values.

```
pml_train <- pml_train[, colSums(is.na(pml_train)) == 0]
pml_test <- pml_test[, colSums(is.na(pml_test)) == 0]
```

As the first 7 columns are extraneous to the accelerometer data which we are going to use for our predictions, they are also removed.

```
pml_train <- pml_train[-c(1:7)]
pml_test <- pml_test[-c(1:7)]
```

A check is then run to ensure that both the training and test set contains the same column names except for the **classe** and **problem_id** columns.

```
all.equal(colnames(pml_train[1:length(pml_train)-1]),colnames(pml_test[1:length(pml_test)-1]))
```

The accelerometer data columns in both the training and test data sets are then all standardized as a numeric variable.

```
for(i in c(1:(ncol(pml_train)-1))) {
  pml_train[,i] = as.numeric(as.character(pml_train[,i]))
  pml_test[,i] = as.numeric(as.character(pml_test[,i]))
}
```

Slicing the Data

In order to help obtain the best model in terms of high Accuracy while minimizing Out of Sample Error, we will further split our training data set into a pure training data set(70%) and a validation data set (30%). A seed is also set to help ensure reproducibility.

```
set.seed(901206)
inTrain <- createDataPartition(pml_train$classe, p=0.70, list=F)
data_train <- pml_train[inTrain, ]
data_cv <- pml_train[-inTrain, ]
```

Model Creation

1. Decision Tree

We will first attempt modeling the data using Decision Trees. 5-fold cross validation would be used when applying the algorithm.

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

We then cross-validate our model to gauge it's accuracy and out of sampling error

```
predictmodelTree <- predict(modelTree, data_cv)
confusionMatrix(predictmodelTree, data_cv$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1502  495  457  443  147
##          B   34  367   39  159  142
##          C  133  277  530  362  316
##          D    0    0    0    0    0
##          E    5    0    0    0  477
##
## Overall Statistics
##
##                Accuracy : 0.4887
##                  95% CI : (0.4759, 0.5016)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3321
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8973  0.32221  0.51657   0.0000  0.44085
## Specificity            0.6338  0.92120  0.77609   1.0000  0.99896
## Pos Pred Value         0.4934  0.49528  0.32756      NaN  0.98963
## Neg Pred Value         0.9395  0.84992  0.88376   0.8362  0.88803
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2552  0.06236  0.09006   0.0000  0.08105
## Detection Prevalence   0.5172  0.12591  0.27494   0.0000  0.08190
## Balanced Accuracy      0.7655  0.62170  0.64633   0.5000  0.71990
```

```
modelTree_accuracy <- postResample(predictmodelTree, data_cv$classe)
modelTree_accuracy
```

```
##  Accuracy     Kappa
## 0.4887001 0.3320768
```

```
modelTree_oose <- 1 - as.numeric(confusionMatrix(data_cv$classe, predictmodelTree)$overall[1])
modelTree_oose
```

```
## [1] 0.5112999
```

The low accuracy of 48.87% was clearly disappointing. This would mean that I could expect a considerable amount of my test-samples to be misclassified.

2. Random Forest

Due to the poor performance of Decision Trees, we shall now attempt to model the data using Random Forests. We should expect Random Forests to perform better than Decision Trees as they are ensembles of decision trees that carries out random record and variable selection to avoid overfitting. Likewise, we shall also be using 5-fold cross validation when applying the algorithm.

```
control <- trainControl(method="cv", 5)
modelForest <- train(classe ~ ., data=data_train, method="rf", trControl=control)
modelForest
```

We then cross-validate our model to gauge it's accuracy and out of sampling error

```
predictmodelForest <- predict(modelForest, data_cv)
confusionMatrix(predictmodelForest, data_cv$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670   13    0    0    0
##          B    2 1123   11    0    0
##          C    1    3 1012    8    2
##          D    0    0    3  955    4
##          E    1    0    0    1 1076
##
## Overall Statistics
##
##                Accuracy : 0.9917
##                  95% CI : (0.989, 0.9938)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9895
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9860   0.9864   0.9907   0.9945
## Specificity            0.9969   0.9973   0.9971   0.9986   0.9996
## Pos Pred Value         0.9923   0.9886   0.9864   0.9927   0.9981
## Neg Pred Value         0.9990   0.9966   0.9971   0.9982   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1908   0.1720   0.1623   0.1828
## Detection Prevalence   0.2860   0.1930   0.1743   0.1635   0.1832
## Balanced Accuracy      0.9973   0.9916   0.9917   0.9946   0.9970
```

```
modelForest_accuracy <- postResample(predictmodelForest, data_cv$classe)
modelForest_accuracy
```

```
##  Accuracy     Kappa
## 0.9916737 0.9894663
```

```
modelForest_oose <- 1 - as.numeric(confusionMatrix(data_cv$classe, predictmodelForest)$overall[1])
modelForest_oose
```

```
## [1] 0.008326253
```

As expected, Random Forests performs a lot better with an accuracy of 99.25% and an Out of Sample Error of 0.75%. As the test set is a sample size of 20, with an accuracy of 99.25%, we can expect that few or none of the test samples will be mis-classified.

Predicting for the Test Data Set

We would now apply our Random Forest model on the original testing data set.

```
result <- predict(modelForest, pml_test[, -length(names(pml_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
```