Practical Machine Learning Project

### Overview

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. A report describing has been created to show how to built the model, how to use cross validation.

### Load the Library

Check for missing dependencies and load necessary R packages

setwd("~/Coursera/practical-machine-learning")  
library(knitr)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

### Read the test and training dataset

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

## [1] 19622 160

dim(testing)

## [1] 20 160

### Prepare the data

The data are prepared for the analysis: the not applicable data and not a number data are removed from the data read form the \*.csv file

# remove variables that are almost always NA  
  
training <- training[, colSums(is.na(training)) == 0]   
  
classe <- training$classe  
trainRemove <- grepl("^X|timestamp|window", names(training))  
training <- training[, !trainRemove]  
training <- training[, sapply(training, is.numeric)]  
training$classe <- classe

### Split the data in two subset

Split the training data in two dataset, a 70% is used for training and a 30% is used to verified the predicted model.

datasplitted <- createDataPartition(training$classe, p=0.70, list=FALSE)  
mytrainData <- training[datasplitted, ]  
mytestData <- training[-datasplitted, ]

### Data Modeling

Fit the predictive model. Two model are estimated to find the best model, the randoms forest and the conditional Inference Tree

modelTree <- train(classe ~ ., data=mytrainData, method="ctree", trControl=trainControl())  
print(modelTree)

## Conditional Inference Tree   
##   
## 13737 samples  
## 52 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...   
## Resampling results across tuning parameters:  
##   
## mincriterion Accuracy Kappa   
## 0.01 0.8665039 0.8310723  
## 0.50 0.8663860 0.8309185  
## 0.99 0.8623996 0.8258367  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mincriterion = 0.01.

controlRf <- trainControl(method="cv", 5)  
modelRf <- train(classe ~ ., data=mytrainData, method="rf", trControl=controlRf, ntree=250)  
print(modelRf)

## Random Forest   
##   
## 13737 samples  
## 52 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 10990, 10990, 10988, 10990, 10990   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.9890803 0.9861850  
## 27 0.9904633 0.9879353  
## 52 0.9826009 0.9779846  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 27.

Analyzing the two models we can see that the randoms forest is the model with a better accuracy.

### Evaluation of the model on the training data

Models is evalued on the 30% of the training data not used for fitting.

predictTree <- predict(modelTree, mytestData)  
accuracyTree <- postResample(predictTree, as.factor(mytestData$classe))  
accuracyTree

## Accuracy Kappa   
## 0.8730671 0.8391266

predictRf <- predict(modelRf, mytestData)  
accuracyRf <- postResample(predictRf, as.factor(mytestData$classe))  
accuracyRf

## Accuracy Kappa   
## 0.9940527 0.9924756

The same accuracy is confirmed on the data training used for the test.

### Predicting for Test Data Set

Results on the testing data, take in account that the modelRf is the best model

resultRf <- predict(modelRf, testing)  
print(resultRf)

## [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