Practical Machine Learing Course Project

Abstract

This report is to predict the manner in which people did the exercise. The data is from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways: 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).

In data modeling, the training data is divided into training dataset and validation dataset. The “classe” variable in the training set is the outcome. The validation data is used to validate the model with the expected out-of-sample error rate of less than 1%, or 99% accuracy. Three data models are used for model predition: Random Forest, Decision Tree and Generalized Boosted Model. The one with highest accuracy is used to model the testing data. The results show that the training model which is build using Random Forest has the highest accurancy (99.29%). Also, it predicts the 20 test cases in the testing data with 100% accuracy.

Data Processing

- Download and unzipped the data to the working directory.
- Set the current working directory to your working directory. For example, use setwd() function.

```
setwd("C:/Users/yyan-koo/Data Science Specification/Courses/Course 8 Machine Le
arning/Week 4/Assignment/Data")
```

Load the data and library

```
library(AppliedPredictiveModeling)
```

```
## Warning: package 'AppliedPredictiveModeling' was built under R version
## 3.2.5
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.2.5
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.2.5
```

```
## Loading required package: rpart
```

```
## Warning: package 'rpart' was built under R version 3.2.5
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

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

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

```
dim(testing)
```

```
## [1]  20 160
```

The training data set contains 19622 observations and 160 variables, while the testing data set contains 20 observations and 160 variables. The “classe” variable in the training set is the outcome to predict.

Data Cleaning

Remove NA values

Compute the prediction only on the accelerometers values of belt, forearm, arm and dumbell. So, remove the following variables.

- Have NA or empty values
- Have nearly zero variance
- Not contribute to the accelerometer measurements

```
# remove variables contain NA missing values.
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]

# remove variables have nearly zero variance
nzv <- nearZeroVar(training)
training <- training[, -nzv]

nzv <- nearZeroVar(testing)
testing <- testing[, -nzv]

# remove variables do not contribute to the accelerometer measurements.
training <- training[, -(1:6)]
testing <- testing[, -(1:6)]

dim(training)
```

```
## [1] 19622    53
```

```
dim(testing)
```

```
## [1] 20 53
```

The cleaned training dataset contains 19622 observations and 53 variables, while the cleaned testing dataset contains 20 observations and 53 variables.

Data Partitioning

Split the cleaned training set into training dataset (70%) and a validation dataset (30%). The validation dataset is used for cross validation.

```
set.seed(123)   # set the seed
inTrain <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
train1 <- training[inTrain, ]
train2 <- training[-inTrain, ]
```

Data Modeling

Random Forest, Decision Tree and Generalized Boosted Model are used for model prediction. The one with highest accuracy will be used to model the testing data.

Random Forest

Estimate the performance of the model on the training dataset.

```
controlRf <- trainControl(method="cv", 5)     # use 5-fold cross validation
modelRf <- train(classe ~ ., data=train1, method="rf", trControl=controlR
f)      # train the model
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: 10989, 10988, 10990, 10991, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9899541  0.9872919
##   27    0.9900272  0.9873849
##   52    0.9858781  0.9821379
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
modelRf$finalModel      # show the final model
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.68%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    3    0    0    1 0.001024066
## B   21 2632    4    1    0 0.009781791
## C    0   12 2374   10    0 0.009181970
## D    0    1   30 2220    1 0.014209591
## E    0    1    5    4 2515 0.003960396
```

Estimate the performance of the model on the validation dataset.

```
predictRf <- predict(modelRf, train2)
confusionMatrix(train2$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    2    0    0    0
##          B    7 1131    1    0    0
##          C    0    8 1015    3    0
##          D    0    0   17  946    1
##          E    0    0    2    1 1079
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9904, 0.9949)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9912   0.9807   0.9958   0.9991
## Specificity            0.9995   0.9983   0.9977   0.9964   0.9994
## Pos Pred Value         0.9988   0.9930   0.9893   0.9813   0.9972
## Neg Pred Value         0.9983   0.9979   0.9959   0.9992   0.9998
## Prevalence             0.2853   0.1939   0.1759   0.1614   0.1835
## Detection Rate         0.2841   0.1922   0.1725   0.1607   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9977   0.9948   0.9892   0.9961   0.9992
```

```
sampleError <- 1 - as.numeric(confusionMatrix(train2$classe, predictRf)$overall
[1])
sampleError
```

```
## [1] 0.007136788
```

The out-of-sample error rate is 0.007136788.

Decision Trees

Estimate the performance of the model on the training dataset.

```
set.seed(123)
modelDT <- rpart(classe ~ ., data=train1, method="class")
```

Estimate the performance of the model on the validation dataset.

```
predictDT <- predict(modelDT, train2, type="class")
confusionMatrix(train2$classe, predictDT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1424   26   33  167   24
##          B  184  667  107  106   75
##          C   52   54  701  193   26
##          D   56   48   53  735   72
##          E   25   70   52  157  778
##
## Overall Statistics
##
##                Accuracy : 0.7315
##                  95% CI : (0.72, 0.7428)
##     No Information Rate : 0.2958
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6606
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8179   0.7711   0.7410   0.5412   0.7979
## Specificity            0.9397   0.9060   0.9342   0.9494   0.9381
## Pos Pred Value         0.8507   0.5856   0.6832   0.7624   0.7190
## Neg Pred Value         0.9247   0.9583   0.9496   0.8734   0.9590
## Prevalence             0.2958   0.1470   0.1607   0.2308   0.1657
## Detection Rate         0.2420   0.1133   0.1191   0.1249   0.1322
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8788   0.8385   0.8376   0.7453   0.8680
```

```
sampleErrorDT <- 1 - as.numeric(confusionMatrix(train2$classe, predictDT)$overa
ll[1])
sampleErrorDT
```

```
## [1] 0.2684792
```

The out-of-sample error rate is 0.2684792.

Generalized Boosted Model

Estimate the performance of the model on the training dataset.

```
set.seed(123)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modelGBM <- train(classe ~ ., data=train1, method = "gbm",
                  trControl = controlGBM, verbose = FALSE)
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.2.5
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

```
## Loading required package: plyr
```

```
modelGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 42 had non-zero influence.
```

Estimate the performance of the model on the validation dataset.

```
predictGBM <- predict(modelGBM, newdata=train2)
confusionMatrix(predictGBM, train2$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1647   52    0    1    6
##          B   13 1057   36    2   10
##          C   10   27  977   41    6
##          D    3    3   11  913   13
##          E    1    0    2    7 1047
##
## Overall Statistics
##
##                Accuracy : 0.9585
##                  95% CI : (0.9531, 0.9635)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9475
##  Mcnemar's Test P-Value : 3.449e-11
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9280   0.9522   0.9471   0.9677
## Specificity            0.9860   0.9871   0.9827   0.9939   0.9979
## Pos Pred Value         0.9654   0.9454   0.9208   0.9682   0.9905
## Neg Pred Value         0.9935   0.9828   0.9898   0.9897   0.9928
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2799   0.1796   0.1660   0.1551   0.1779
## Detection Prevalence   0.2899   0.1900   0.1803   0.1602   0.1796
## Balanced Accuracy      0.9849   0.9576   0.9675   0.9705   0.9828
```

```
sampleErrorGBM <- 1 - as.numeric(confusionMatrix(train2$classe, predictGBM)$ove
rall[1])
sampleErrorGBM
```

```
## [1] 0.04146134
```

The out-of-sample error rate is 0.04146134.

Conclusion

The accuracy of the 3 data models are:

- Random Forest : 0.9929
- Decision Tree : 0.7315
- Generalized Boosted Model : 0.9585

Among the three data modeling, Random Forest has the highest accurancy. Therefore, it is used to predit the testing data.

Predicting for Test Data Set

Apply the model to the original testing data downloaded from the data source. First, remove the problem_id column .

```
result <- predict(modelRf, testing[, -length(names(testing))])
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
```

The result of model predition is B A B A A E D B A A B C B A E E A B B B.

Create function to write predictions to files

```
pml_write_files = function(x){
  n = length(x)
  path <- "C:/Users/yyan-koo/Data Science Specification/Courses/Course 8 Machin
e Learning/Week 4/Assignment/Data"
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=file.path(path, filename),quote=FALSE,row.names=FALS
E,col.names=FALSE)
  }
}
# create prediction files
pml_write_files(result)
```