Practical Machine Learning Model Project

Karlo dela Cruz

November 16, 2018

Executive Summary

In this case study, the objective is to predict the manner in which they did the exercise (classe variable) using the data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har).

For this report, I employed 3 models: CART, CART with 10 fold Cross-Validation, and Random Forest. Among the predictive models, the Random Forest Model has the best accuracy of 99.61%.

Then, the final model is applied to a test set of 20 samples and correctly predicts the outcome variable.

Data Preparation

Load data of the training data for predictive model building and testing data of 20 samples to correctly predicts the outcome variables.

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

For the training data, here we need to split the data to train and test data for model building.

```
library(caret)
```

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

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

```
inBuild <- createDataPartition(y=training$classe,
                               p=0.7, list=FALSE)
train <- training[inBuild,]
test <- training[-inBuild,]
```

Remove zero covariates; Reduced 160 variables to 104 variables

```
nsv <- nearZeroVar(train,saveMetrics=FALSE)
train <- train[,-nsv]
test <- test[,-nsv]
```

Remove variables with more than 5% missing values; Reduced 104 variables to 59 variables

```
miss <- which(colMeans(is.na(train)) >= 0.05)
train <- train[,-miss]
test <-test[,-miss]
```

Focus on the data from accelerometer; Reduced 59 variables to 54 variables.

```
train <- train[,-(1:5)]
test <- test[,-(1:5)]
```

Machine Learning Models (Classification Prediction)

For CART (rpart), the model was able to predict the outcome variable with 73.47% Accuracy. Simple decision tree is developed.

```
set.seed(123)
library(rpart)
rpartmod <- rpart(classe ~ .,data=train)
predrpart <- predict(rpartmod, newdata = test, type = "class")
confusionMatrix(test$classe, predrpart)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1539   39   37   50    9
##          B  150  684  145  129   31
##          C   46   44  914   19    3
##          D   53   24  188  600   99
##          E   10   34   92   88  858
##
## Overall Statistics
##
##                Accuracy : 0.7808
##                  95% CI : (0.77, 0.7913)
##     No Information Rate : 0.3055
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7223
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8560   0.8291   0.6642   0.6772   0.8580
## Specificity            0.9670   0.9101   0.9752   0.9272   0.9541
## Pos Pred Value         0.9194   0.6005   0.8908   0.6224   0.7930
## Neg Pred Value         0.9385   0.9703   0.9049   0.9419   0.9704
## Prevalence             0.3055   0.1402   0.2338   0.1506   0.1699
## Detection Rate         0.2615   0.1162   0.1553   0.1020   0.1458
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9115   0.8696   0.8197   0.8022   0.9061
```

For CART with 10-fold cross-validation, the model was able to predict the outcome variable with 95.45% Accuracy. For this case, complexity parameter (cp) of the decision tree model should be determined. Results show that the cp should be 0.0002. The cp parameter is considered in the CART model.

```
library(caret)
numFolds <- trainControl(method = "cv", number = 10)
cpGrid <- expand.grid(.cp = seq(0.0001,0.001,0.0001))
train(classe ~ ., data = train, method = "rpart", trControl = numFolds, tuneGrid = cpGrid)
```

```
## CART
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12363, 12364, 12362, 12363, 12366, 12362, ...
## Resampling results across tuning parameters:
##
##   cp     Accuracy   Kappa
##   1e-04  0.9575586  0.9463247
##   2e-04  0.9575589  0.9463248
##   3e-04  0.9566121  0.9451281
##   4e-04  0.9547916  0.9428300
##   5e-04  0.9531172  0.9407040
##   6e-04  0.9506426  0.9375714
##   7e-04  0.9478029  0.9339719
##   8e-04  0.9441621  0.9293644
##   9e-04  0.9429972  0.9278903
##   1e-03  0.9385569  0.9222749
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 2e-04.
```

```
rpartmod <- rpart(classe ~ .,data=train, cp = 0.0002)
predrpart <- predict(rpartmod, newdata = test, type = "class")
confusionMatrix(test$classe, predrpart)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1652   12    4    3    3
##          B   21 1074   28    9    7
##          C    0    5 1005   10    6
##          D    4   29   15  903   13
##          E    1   19    6   21 1035
##
## Overall Statistics
##
##                Accuracy : 0.9633
##                  95% CI : (0.9582, 0.968)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9536
##  Mcnemar's Test P-Value : 5.923e-06
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9845   0.9429   0.9499   0.9545   0.9727
## Specificity            0.9948   0.9863   0.9956   0.9876   0.9903
## Pos Pred Value         0.9869   0.9429   0.9795   0.9367   0.9566
## Neg Pred Value         0.9938   0.9863   0.9891   0.9913   0.9940
## Prevalence             0.2851   0.1935   0.1798   0.1607   0.1808
## Detection Rate         0.2807   0.1825   0.1708   0.1534   0.1759
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9896   0.9646   0.9728   0.9711   0.9815
```

For Random Forest, the model was able to predict the outcome variable with 99.61% Accuracy. In this model, the model generated 200 decsion trees and minimum of 5 samples per rules of the decision tree.

```
library(randomForest)
```

```
## randomForest 4.6-14
```

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

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

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

```
set.seed(123)
rfmod <- randomForest(classe ~ .,data=train, ntree = 200, nodesize = 5)
predrf <- predict(rfmod, newdata = test)
confusionMatrix(test$classe, predrf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    4 1133    2    0    0
##          C    0    0 1022    4    0
##          D    0    0    1  963    0
##          E    0    2    0    3 1077
##
## Overall Statistics
##
##                Accuracy : 0.9973
##                  95% CI : (0.9956, 0.9984)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9966
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9982   0.9971   0.9928   1.0000
## Specificity            1.0000   0.9987   0.9992   0.9998   0.9990
## Pos Pred Value         1.0000   0.9947   0.9961   0.9990   0.9954
## Neg Pred Value         0.9991   0.9996   0.9994   0.9986   1.0000
## Prevalence             0.2851   0.1929   0.1742   0.1648   0.1830
## Detection Rate         0.2845   0.1925   0.1737   0.1636   0.1830
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9988   0.9985   0.9981   0.9963   0.9995
```

Conclusion

In this case study, we'll predict the manner in which they did the exercise (classe variable) using random forest prediction algorithm since it has the best accuracy of 99.61% among the 3 classification models. Then, the model is applied to a testing set of 20 samples.

```
predfinal <- predict(rfmod, newdata = testing)
predfinal
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```