Prediction

Connie

November 18, 2017

1. Download the datasets

```
setwd("C:/Users/conni/Desktop/R Projects/ML")

train <- read.csv("pml-training.csv", na.strings = c("NA", "#DIV/0!", ""))
test <- read.csv("pml-testing.csv",  na.strings = c("NA", "#DIV/0!", ""))

dim(train)
```

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

```
dim(test)
```

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

The train dataset has 19622 observations and 160 variables and the test data set contains 20 observations and 160 variables. The goal of the project is to predict the outcome of the variable "Classe" in the test dataset.

2. Clean the datasets

Removing the first 7 columns from the data as they have little predicting power and delete columns (predictors) of the training set that contain any missing values.

```
train <- train[, 8:160]
test  <- test[, 8:160]

train <- train[, colSums(is.na(train)) == 0]
test <- test[, colSums(is.na(test)) == 0]

dim(train)
```

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

```
dim(test)
```

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

After cleaning, both dataset's columns reduce to 53.

3. Split the train dataset into two sets

train1 is the training data set (it contains 11776 rows or 60% of the entire train data set), train2 is the testing data set (it contains 7846 rows or 40% of the entire train data set)

```
library(caret)

set.seed(12345)
```

```
inTrain <- createDataPartition(y=train$classe, p=0.60, list=FALSE)
train1  <- train[inTrain,]
train2  <- train[-inTrain,]
dim(train1)
```

```
## [1] 11776    53
```

```
dim(train2)
```

```
## [1] 7846   53
```

4. Prediction Model Building

Using random forests

```
library(randomForest)

set.seed(12345)
control <- trainControl(method="cv", number=5, verboseIter=FALSE)
modFitRF <- train(classe ~ ., data=train1, method="rf",
                          trControl=control)
modFitRF
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9421, 9421, 9422, 9420, 9420
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9901496  0.9875387
##   27    0.9880266  0.9848530
##   52    0.9847997  0.9807700
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
predict_rf <- predict(modFitRF,train2)
conf_rf <- confusionMatrix(train2$classe, predict_rf)
conf_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    2    0    0    0
##          B    9 1505    4    0    0
##          C    0    9 1359    0    0
##          D    0    0   20 1264    2
```

```
##          E    0    0    2    3 1437
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9915, 0.9952)
##     No Information Rate : 0.2854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9918
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9960   0.9927   0.9812   0.9976   0.9986
## Specificity            0.9996   0.9979   0.9986   0.9967   0.9992
## Pos Pred Value         0.9991   0.9914   0.9934   0.9829   0.9965
## Neg Pred Value         0.9984   0.9983   0.9960   0.9995   0.9997
## Prevalence             0.2854   0.1932   0.1765   0.1615   0.1834
## Detection Rate         0.2842   0.1918   0.1732   0.1611   0.1832
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9978   0.9953   0.9899   0.9971   0.9989
```

The out-of-sample error rate is 1-0.9935= 0.65%

Using classification trees

```
library(rpart)
modFitRPT <- rpart(classe ~ ., data=train1, method="class")
predict_rpt <- predict(modFitRPT, train2, type = "class")
conf_rpart <- confusionMatrix(train2$classe, predict_rpt)
conf_rpart
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1879   56  105  155   37
##          B  260  759  340  132   27
##          C   30   88 1226   23    1
##          D   69   34  354  807   22
##          E   66   54  234   57 1031
##
## Overall Statistics
##
##                Accuracy : 0.7267
##                  95% CI : (0.7167, 0.7366)
##     No Information Rate : 0.2937
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6546
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8155  0.76589   0.5427   0.6874   0.9222
## Specificity            0.9363  0.88928   0.9746   0.9282   0.9389
## Pos Pred Value         0.8418  0.50000   0.8962   0.6275   0.7150
## Neg Pred Value         0.9243  0.96334   0.8405   0.9441   0.9864
## Prevalence             0.2937  0.12631   0.2879   0.1496   0.1425
## Detection Rate         0.2395  0.09674   0.1563   0.1029   0.1314
## Detection Prevalence   0.2845  0.19347   0.1744   0.1639   0.1838
## Balanced Accuracy      0.8759  0.82759   0.7587   0.8078   0.9305
```

The out-of-sample error rate is 1-0.7267= 27.33%

Conclusion:

For this dataset random forest method is better than classification tree method. Random forest has accuracy rate 0.9935 and out-of-sample error rate is 0.0065. Classification tree has accuracy rate 0.7267 and out-of-sample error rate is 0.2733.

5.For Predictions on test dataset

```
predict(modFitRF, test)
```

```
##  [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
```