Practical Machine Learning Projcet

neelani

May 28, 2017

Project Background

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. In this project, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. This exercise is to predict the barbell lifts done 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

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 come from this source: http://groupware.les.inf.puc-rio.br/har.

Model Design Approach

To determine the best prediction model, it will be always better to build few models with different configurations and pick the best one. In this exercise, we will be building Linear Discriminant Analysis(LDA), Decision Tree and Random Forest Model to select the best one.

Load the required libraries

```
library(foreach)
library(lattice)
library(ggplot2)
library(MASS)
library(rpart)
library(parallel)
library(iterators)
library(doParallel)
library(caret)
library(randomForest)

## 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
```

Set the environment for parallelism

```
## Clear the Global Environment
rm(list=ls())
wd <- getwd()

# Calculate the number of cores to run the process in parallel mode
no_cores <- detectCores() - 1
```

Download the required files

```
downloadFile <- "pml_training.csv"
if (!file.exists(downloadFile)) {
  url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
  download.file(url, destfile = downloadFile)
}
trainData <- read.csv(downloadFile, na.strings = c("NA","#DIV/0!",""))

downloadFile <- "pml_testing.csv"
if (!file.exists(downloadFile)) {
  url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
  download.file(url, destfile = downloadFile)
}
testData <- read.csv(downloadFile, na.strings = c("NA","#DIV/0!",""))
```

Cleanup the training data by removing NA, near zero and no-value-add attributes

```
trainData <- trainData[, -(1:7)]
testData  <- testData[, -(1:7)]

## Remove near zero value columns
nzv <- nearZeroVar(trainData, saveMetrics = TRUE)
trainData <- trainData[, -nzv$nzv == FALSE]

# remove variables that are almost always NA
nas <- sapply(trainData, function(x) mean(is.na(x))) > 0.95
trainData <- trainData[, nas==FALSE]


## Align the columns of test data similar to train data set
```

```
cnames <- colnames(trainData)
testData <- testData[, cnames[-53]]
```

Split the training set data for cross validation

```
# Divide the training data into a two sets for trainig and cross validation
set.seed(12345)
splitData <- createDataPartition(trainData$classe, p=0.7,list=FALSE)
trainDataSet1 <- as.data.frame(trainData[splitData,])
trainDataSet2 <- as.data.frame(trainData[-splitData,])

# Set the trainControl parameters for the 5 fold cross validation
control <- trainControl(method = "cv", number = 5)
```

Build Linear Discriminant Analysis (LDA) Model

Build Linear Discriminant Analysis (LDA) Model, apply the LDA to trainingDataSet1 and print the confusion matrix

```
# Create the Linear Discriminant Analysis (LDA) in Paralel mode

# Initiate cluster
cl <- makeCluster(no_cores)
registerDoParallel(cl)

ldaModelFit <- train(classe ~ .,
                     data = trainDataSet1,
                     method = "lda",
                     trcontrol = control)
stopCluster(cl)


# Save the LDA Model
save(ldaModelFit, file = "LDA_modelFit.RData")

# Apply the LDA to training Data Set 1 and validate it using confusion matrix
predictSet1 <- predict(ldaModelFit, trainDataSet1)
confusionMatrix(predictSet1, trainDataSet1$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3207  402  259  146   95
##          B   81 1719  228   82  414
##          C  294  307 1585  250  241
##          D  313  110  273 1683  233
##          E   11  120   51   91 1542
##
## Overall Statistics
##
```

```
##                Accuracy : 0.7087
##                  95% CI : (0.7011, 0.7163)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6313
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8210   0.6467   0.6615   0.7473   0.6107
## Specificity            0.9082   0.9273   0.9037   0.9191   0.9757
## Pos Pred Value         0.7805   0.6811   0.5921   0.6443   0.8496
## Neg Pred Value         0.9274   0.9163   0.9267   0.9489   0.9175
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2335   0.1251   0.1154   0.1225   0.1123
## Detection Prevalence   0.2991   0.1837   0.1949   0.1901   0.1321
## Balanced Accuracy      0.8646   0.7870   0.7826   0.8332   0.7932
```

Build Desision Tree Model

Build Desision Tree Model, apply the model to trainingDataSet1 and print the confusion matrix

```
# Create the Decision Tree Model in Paralel mode

# Initiate cluster
cl <- makeCluster(no_cores)
registerDoParallel(cl)

dtModelFit <- train(classe ~ .,
                    data = trainDataSet1,
                    method = "rpart")
stopCluster(cl)


# Save the Decision Tree Model
save(dtModelFit, file = "DecisionTree_modelFit.RData")

# Apply the Decision Tree Model to training Data Set 1 and validate it using
confusion matrix
predictSet1 <- predict(dtModelFit, trainDataSet1)
confusionMatrix(predictSet1, trainDataSet1$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3526 1111 1120 1011  350
```

```
##          B   60  906   79  384  319
##          C  277  641 1197  857  674
##          D    0    0    0    0    0
##          E   43    0    0    0 1182
##
## Overall Statistics
##
##                Accuracy : 0.4958
##                  95% CI : (0.4874, 0.5042)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3412
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9027  0.34086  0.49958   0.0000  0.46812
## Specificity            0.6346  0.92400  0.78406   1.0000  0.99616
## Pos Pred Value         0.4954  0.51831  0.32830      NaN  0.96490
## Neg Pred Value         0.9426  0.85387  0.88118   0.8361  0.89266
## Prevalence             0.2843  0.19349  0.17442   0.1639  0.18381
## Detection Rate         0.2567  0.06595  0.08714   0.0000  0.08604
## Detection Prevalence   0.5182  0.12725  0.26541   0.0000  0.08918
## Balanced Accuracy      0.7687  0.63243  0.64182   0.5000  0.73214
```

Build Random Forest Model

Build Random Forest Model, apply the model to trainingDataSet1

```
# Create the Random Forest Model in Paralel mode

modelFile <- "RandonForest_modelFit.RData"

if (!file.exists(modelFile)) {
  # Initiate cluster
  cl <- makeCluster(no_cores)
  registerDoParallel(cl)

  rfModelFit <- train(classe ~ .,
                      data = trainDataSet1,
                      method = "rf",
                      trcontrol = control)
  stopCluster(cl)

  # Save the Random Forest Model
  save(rfModelFit, file = "RandonForest_modelFit.RData")
} else {
  rfModelFit <-get(load(paste(wd, modelFile, sep = "/")))
```

```
}

# Apply the Random Forest Model to training Data Set 1 and validate it using
confusion matrix
predictSet1 <- predict(rfModelFit, trainDataSet1)
confusionMatrix(predictSet1, trainDataSet1$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3901   12    0    0    1
##          B    4 2645    4    0    1
##          C    0    1 2381   19    5
##          D    0    0   11 2231    7
##          E    1    0    0    2 2511
##
## Overall Statistics
##
##                Accuracy : 0.995
##                  95% CI : (0.9937, 0.9962)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9937
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9951   0.9937   0.9907   0.9945
## Specificity            0.9987   0.9992   0.9978   0.9984   0.9997
## Pos Pred Value         0.9967   0.9966   0.9896   0.9920   0.9988
## Neg Pred Value         0.9995   0.9988   0.9987   0.9982   0.9988
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1925   0.1733   0.1624   0.1828
## Detection Prevalence   0.2849   0.1932   0.1751   0.1637   0.1830
## Balanced Accuracy      0.9987   0.9971   0.9958   0.9946   0.9971
```

Since Random Forest model is predicting with higher accuracy then other models, lets predict trainingDataSet2 to validate the results

```
# Apply the Random Forest Model to training Data Set 2 and validate it using
confusion matrix
predictSet2 <- predict(rfModelFit, trainDataSet2)
confusionMatrix(predictSet2, trainDataSet2$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1670    6    0    0    1
##          B    4 1132    2    0    0
##          C    0    1 1019    9    1
##          D    0    0    5  954    4
##          E    0    0    0    1 1076
##
## Overall Statistics
##
##                Accuracy : 0.9942
##                  95% CI : (0.9919, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9927
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9939   0.9932   0.9896   0.9945
## Specificity            0.9983   0.9987   0.9977   0.9982   0.9998
## Pos Pred Value         0.9958   0.9947   0.9893   0.9907   0.9991
## Neg Pred Value         0.9990   0.9985   0.9986   0.9980   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1924   0.1732   0.1621   0.1828
## Detection Prevalence   0.2850   0.1934   0.1750   0.1636   0.1830
## Balanced Accuracy      0.9980   0.9963   0.9955   0.9939   0.9971
```

Conclusion

Out of the three models, Random Forest model prediction is the best with 99% of accuracy, sensitivity and specificity. Use the Random Forest model to predict the give test data

```
## Test the model with the given test data
predictTestData <- predict(rfModelFit, testData)
predictTestData
```

```
##  [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
```

```
# create function to write predictions to files
write_files <- function(x) {
  n <- length(x)
  for(i in 1:n) {
    filename <- paste0("problem_id_", i, ".txt")
    write.table(x[i], file=filename, quote=F, row.names=F, col.names=F)
  }
}
```

```
# create prediction files to submit
write_files(predictTestData)
```