Activity Recognition Using Predictive Analysis

Created By : Rishab Raina

Introduction

Using devices such as Jawbone Up, Nike Fuel Band, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerators on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: [http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har] (see the section on the Weight Lifting Exercise Data set).

For the purpose of this project, the following steps would be followed:

1. Data Loading
2. Data Cleaning & Preprocessing
3. Exploratory Data Analysis
4. Prediction Model Selection
5. Predicting Test Set Output

Loading the Dataset & Libraries

The Data set has been downloaded from the internet and has been loaded into two seperate dataframes, **"training and "testing"**. The **"training** data set has 19622 number of records and the **"testing** data set has 20 records. The number of variables is 160.

```
# Package names
packages <- c("caret","corrplot","rpart","rpart.plot","RColorBrewer","RGtk2","rattle","randomForest","gb

# Install packages not yet installed
installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  install.packages(packages[!installed_packages])
}

# Packages loading
invisible(lapply(packages, library, character.only = TRUE))

# Loading Dataset using URL
train_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

```
init_training_data <- read.csv(url(train_url))
init_testing_data <- read.csv(url(test_url))
```

Data Cleaning

First, we load the training and testing set from the online sources and then split the training set further into training and test sets.

Removing Variables which have nearly zero variance

```
non_zero_var <- nearZeroVar(init_training_data)

training_data <- init_training_data[,-non_zero_var]
testing_data <- init_testing_data[,-non_zero_var]
```

Removing Variables which are having NA values. Our threshold is 95%.

```
na_val_col <- sapply(training_data, function(x) mean(is.na(x))) > 0.95

training_data <- training_data[,na_val_col == FALSE]
testing_data <- testing_data[,na_val_col == FALSE]
```

Removing variables which will not to used for model building

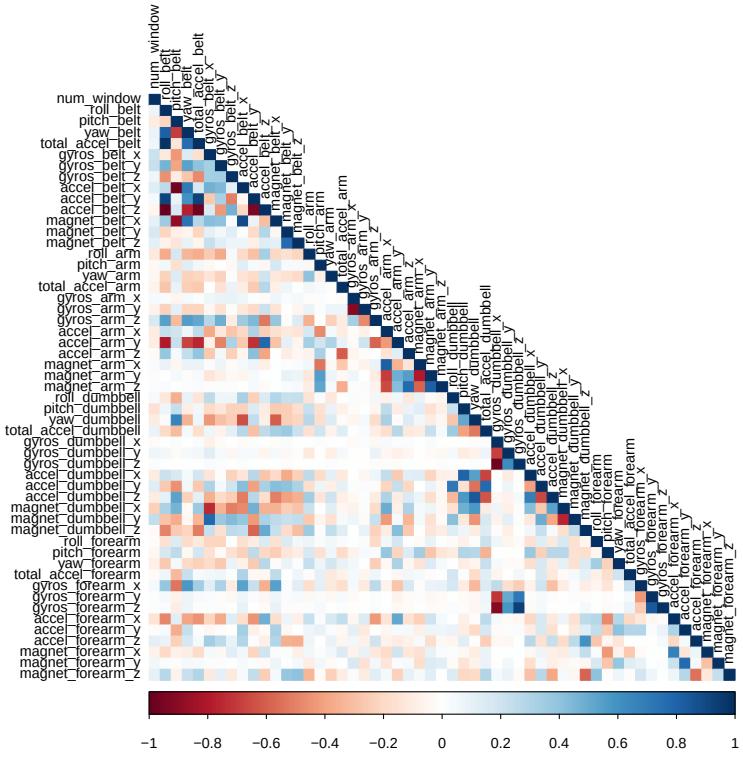
```
training_data <- training_data[,-(1:5)]
testing_data <- testing_data[,-(1:5)]
```

Data Cleaning Complete. As a result of data processing and cleaning, we are able to reduce the variables from 160 to 54

EDA

Now that we have filtered out variables we are going to use in the model, we shall look at the dependence of these variables on each other through a correlation plot.

```
library(corrplot)
corrMat <- cor(training_data[,-54])
corrplot(corrMat, method = "color", type = "lower", tl.cex = 0.8, tl.col = rgb(0,0,0))
```



Data Partitioning

Using the training set, we will be further splitting it into 2 sets - one for training and the other test set for validation to choose the best model. That model would be chosen to run on the actual testing dataset which we left out

```
inTrain <- createDataPartition(training_data$classe, p=0.6, list=FALSE)
training <- training_data[inTrain,]
testing <- training_data[-inTrain,]
```

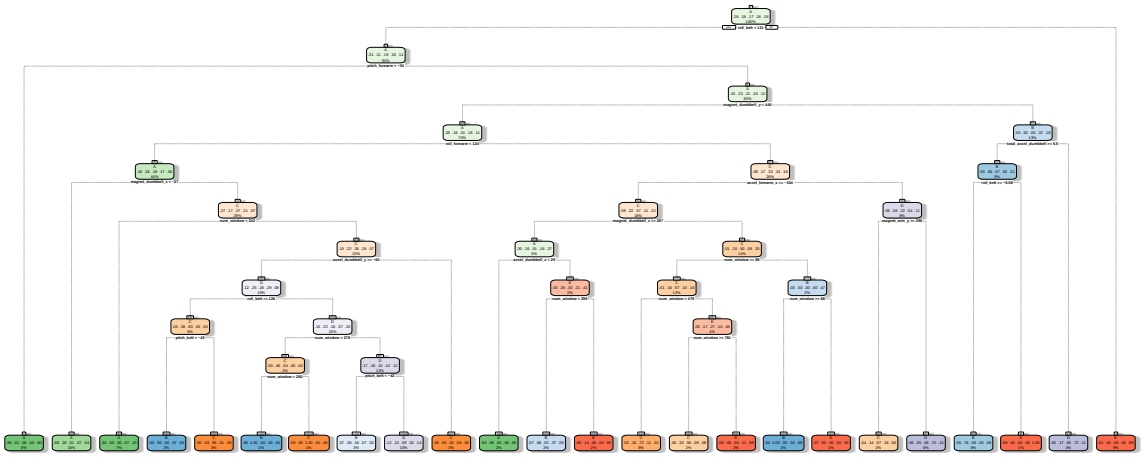
Prediction Model Selection

We will use 3 methods to model the training set and thereby choose the one having the best accuracy to predict the outcome variable in the testing set. The methods are Decision Tree, Random Forest and Generalized Boosted Model.

Note : A confusion matrix plotted at the end of each model will help visualize the analysis better.

Decision Tree

```
set.seed(7867)
modelDT <- rpart(classe ~ ., data = training, method = "class")
fancyRpartPlot(modelDT)
```



Rattle 2021−Sep−10 16:41:16 rishab.raina

```
predictDT <- predict(modelDT, testing, type = "class")
confMatDT <- confusionMatrix(predictDT, as.factor(testing$classe))
confMatDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1942  279   25  133   87
##          B  130  860   78   89   59
##          C   22  194 1183  135   35
##          D  130  172   82  912  197
##          E    8   13    0   17 1064
##
## Overall Statistics
##
##                Accuracy : 0.7598
##                  95% CI : (0.7501, 0.7692)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6954
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8701   0.5665   0.8648   0.7092   0.7379
## Specificity            0.9067   0.9437   0.9404   0.9114   0.9941
## Pos Pred Value         0.7875   0.7072   0.7540   0.6109   0.9655
```

```
## Neg Pred Value         0.9461   0.9008   0.9705   0.9411   0.9440
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2475   0.1096   0.1508   0.1162   0.1356
## Detection Prevalence   0.3143   0.1550   0.2000   0.1903   0.1405
## Balanced Accuracy      0.8884   0.7551   0.9026   0.8103   0.8660
```

Random Forest

```
library(caret)
set.seed(13908)
control <- trainControl(method = "cv", number = 3, verboseIter=FALSE)
modelRF <- train(classe ~ ., data = training, method = "rf", trControl = control, ntree =100)
modelRF$finalModel
```

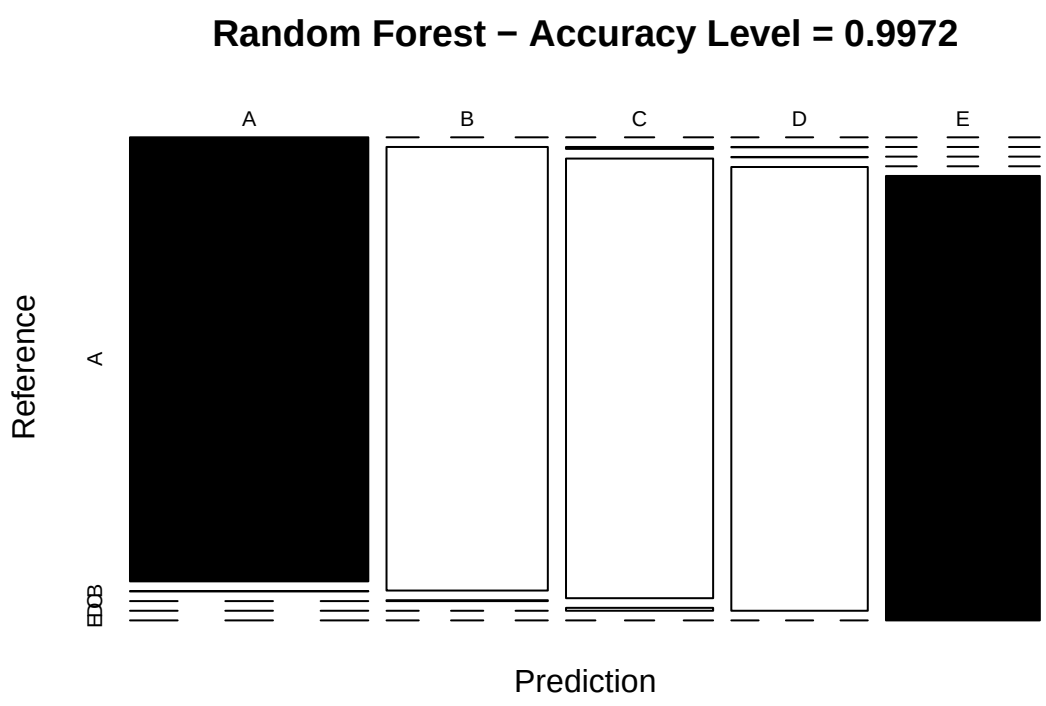
```
##
## Call:
##  randomForest(x = x, y = y, ntree = 100, mtry = min(param$mtry,      ncol(x)))
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.31%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3346    1    0    0    1 0.0005973716
## B    7 2268    4    0    0 0.0048266784
## C    0    7 2046    1    0 0.0038948393
## D    0    0   10 1919    1 0.0056994819
## E    0    1    0    4 2160 0.0023094688
```

```
predictRF <- predict(modelRF, testing)
confMatRF <- confusionMatrix(predictRF, as.factor(testing$classe))
confMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    2    0    0    0
##          B    0 1509    3    0    0
##          C    0    6 1364    9    0
##          D    0    1    1 1277    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9972
##                  95% CI : (0.9958, 0.9982)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.9965
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9941   0.9971   0.9930   1.0000
## Specificity            0.9996   0.9995   0.9977   0.9997   1.0000
## Pos Pred Value         0.9991   0.9980   0.9891   0.9984   1.0000
## Neg Pred Value         1.0000   0.9986   0.9994   0.9986   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1923   0.1738   0.1628   0.1838
## Detection Prevalence   0.2847   0.1927   0.1758   0.1630   0.1838
## Balanced Accuracy      0.9998   0.9968   0.9974   0.9963   1.0000
```

```
plot(confMatRF$table, col = confMatRF$byClass,
     main = paste("Random Forest - Accuracy Level =",
                  round(confMatRF$overall['Accuracy'], 4)))
```



From the Confusion Matrix, we can clearly see that the prediction accuracy of Random Forest model is 99% which is satisfactory.

Generalized Boosted Model

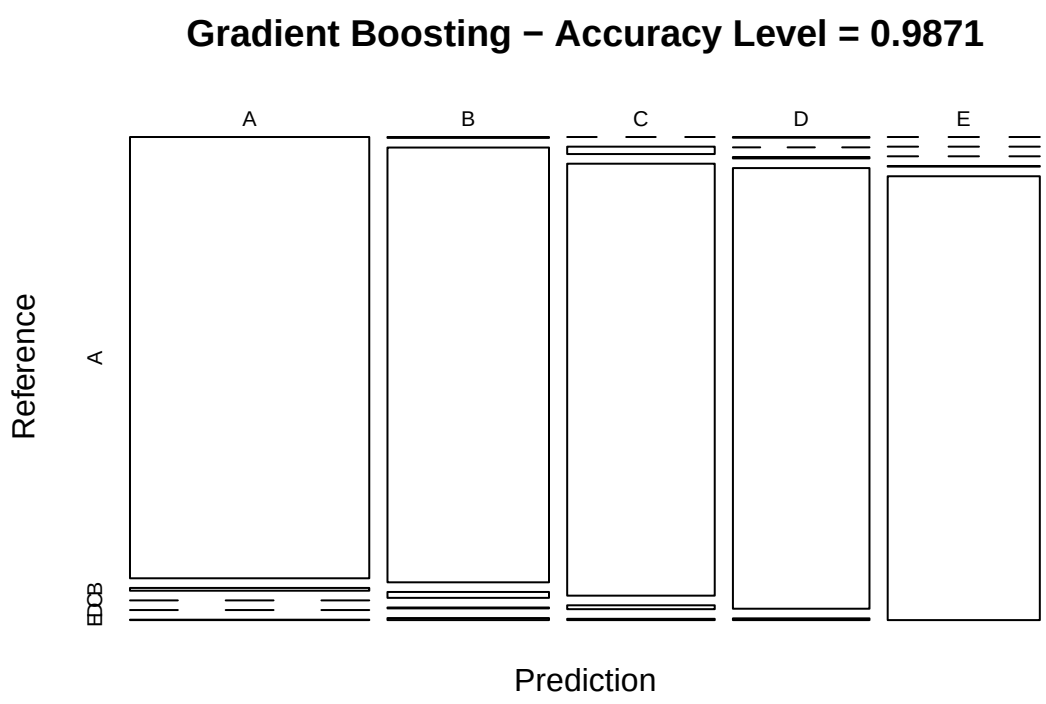
```
library(caret)
set.seed(13908)
control <- trainControl(method = "repeatedcv", number = 5, repeats = 1, verboseIter = FALSE)
modelGBM <- train(classe ~ ., data = training, trControl = control, method = "gbm", verbose = FALSE)
modelGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
predictGBM <- predict(modelGBM, testing)
confMatGBM <- confusionMatrix(predictGBM, as.factor(testing$classe))
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2227   14    0    0    2
##          B    3 1481   20    3    7
##          C    0   23 1344   12    4
##          D    2    0    4 1269    5
##          E    0    0    0    2 1424
##
## Overall Statistics
##
##                Accuracy : 0.9871
##                  95% CI : (0.9844, 0.9895)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9837
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9756   0.9825   0.9868   0.9875
## Specificity            0.9971   0.9948   0.9940   0.9983   0.9997
## Pos Pred Value         0.9929   0.9782   0.9718   0.9914   0.9986
## Neg Pred Value         0.9991   0.9942   0.9963   0.9974   0.9972
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2838   0.1888   0.1713   0.1617   0.1815
## Detection Prevalence   0.2859   0.1930   0.1763   0.1631   0.1817
## Balanced Accuracy      0.9975   0.9852   0.9882   0.9926   0.9936
```

```
plot(confMatGBM$table, col = confMatGBM$byClass,
     main = paste("Gradient Boosting - Accuracy Level =",
                  round(confMatGBM$overall['Accuracy'], 4)))
```



From Gradient Boost Model, the prediction accuracy is 98.7% which is satisfactory.

Now we need to see how each model has predicted the validation dataset across the classifications. We are not considering Decision Tree model as it didn't reach the satisfactory prediction accuracy level. So only Random Forest and Gradient Boosting methods are being compared. As Random Forest offers the maximum accuracy of 99.6%, we will go with Random Forest Model to predict our test data class variable.

Predicting Test Set Output

```
predictRF <- predict(modelRF, testing_data)
```

The final prediction are as follows:

```
##  [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
```