Title: “Machine learning Project”

Author: “Abdou K. Allayeh”

Synopsis

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. 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, we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

The data consists of a Training data and a Test data (to be used to validate the selected model).

The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with.

Data Processing

Preparation of libraries and packages

This step for loading some required packages and initializing some variables

```
install.packages("e1071")
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(randomForest)
library(corrplot)
library(lattice)
library(ggplot2)
library(rattle)
knitr::opts_chunk$set(echo = TRUE)
```

Download the Data

```
setwd("~/RDataScience/ML Project")
training_Url <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml
```

```
-training.csv"
testing_Url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml
-testing.csv"
trainFile <- "~/RDataScience/ML Project/data/pml-training.csv"
testFile  <- "~/RDataScience/ML Project/data/pml-testing.csv"
if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists(trainFile)) {
  download.file(training_Url, destfile=trainFile, method="curl")
}
if (!file.exists(testFile)) {
  download.file(testing_Url, destfile=testFile, method="curl")
}
```

Splitting the Data and Cleaning

Splitting the data into training and testing data. Some basic transformations and cleanup will be performed. We will remove the variables that contains missing values. The pml-training.csv data is used to devise training and testing sets. The pml-test.csv data is used to predict and answer the 20 questions based on the trained model.

```
training_Raw <- read.csv("./data/pml-training.csv")
testing_Raw <- read.csv("./data/pml-testing.csv")
dim(training_Raw)
dim(testing_Raw)
```

##[1] 19622 160

##[1] 20 160

##Note: There are 19622 observations and 160 variables ##Next: Split the data into training data set (70%) and cleaned testing data set (30%)

```
set.seed(10)
library(caret)
library(lattice)
library(ggplot2)
samples <- createDataPartition(y=training_Raw$classe, p=0.7, list=F)
ptrain1 <- training_Raw[samples, ]
ptrain2 <- training_Raw[-samples, ]
dim(ptrain1)
dim(ptrain2)

## Next: Cleaning Step
1- Remove columns that contains NA
2- Remove variables that are NA
3- Remove the Five variables that have no significant impact on the out
come
```

```{r}
nzv <- nearZeroVar(ptrain1)
ptrain1 <- ptrain1[, -nzv]
ptrain2 <- ptrain2[, -nzv]

mostlyNA <- sapply(ptrain1, function(x) mean(is.na(x))) > 0.95
ptrain1 <- ptrain1[, mostlyNA==F]
ptrain2 <- ptrain2[, mostlyNA==F]

ptrain1 <- ptrain1[, -(1:5)]
ptrain2 <- ptrain2[, -(1:5)]

dim(ptrain1)
dim(ptrain2)
```

##[1] 13737 54

##[1] 5885 54

Results

Note: After this step we are down now to 54 variables

The cleaned training data set contains 13737 observations and 54 variables

```
library(corrplot)
cor_mat <- cor(ptrain1[, -54])
corrplot(cor_mat, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```

To obtain the names of the variables we do the following; we use the find Correlation function to search for highly correlated attributes with a cut off equal to 0.70

```
highlyCorrelated = findCorrelation(cor_mat, cutoff=0.70)
names(ptrain1)[highlyCorrelated]
```

#[1] “accel_belt_z” “roll_belt” “accel_belt_y” “accel_arm_y”
#[5] “total_accel_belt” “yaw_belt” “accel_dumbbell_z” “accel_belt_x”
#[9] “pitch_belt” “magnet_dumbbell_x” “accel_dumbbell_y” “magnet_dumbbell_y”
#[13] “accel_arm_x” “accel_dumbbell_x” “accel_arm_z” “magnet_arm_y”
#[17] “magnet_belt_z” “accel_forearm_y” “gyros_dumbbell_y” “gyros_forearm_y”
#[21] “gyros_dumbbell_x” “gyros_dumbbell_z” “gyros_arm_x”

Model Building

Prediction model: Random forest or decision tree will applied to predict the outcomes

1- Random Forest

-Using 3-fold CV to select optimal tuning parameters -Fit model on training data -Print final model to see tuning parameters

```
library(MASS)
library(caret)
library(randomForest)
rfNews()
fitControl <- trainControl(method="cv", number=3, verboseIter=F)
fit <- train(classe ~ ., data=ptrain1, method="rf", trControl=fitContro
l)
fit$finalModel
```

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: 0.22%
```

Confusion matrix: A B C D E class.error A 3904 1 0 0 1 0.0005120328 B 6 2649 3 0 0 0.0033860045 C 0 7 2389 0 0 0.0029215359 D 0 0 6 2245 1 0.0031083481 E 0 0 0 5 2520 0.0019801980

RF Classification into 500 trees and 27 Splits ## Evaluate and validate the model obtained

```
library(caret)
preds <- predict(fit, newdata=ptrain2, call. = FALSE)
CMRF <- confusionMatrix(preds, ptrain2$classe, )
CMRF
```

Results For RF Model:

The accuracy of RF is 0.998 and the out-of-sample-error is 0.002 The best results for RF model, So we will use it for the validation step

Re-training the model

```
nzv <- nearZeroVar(training_Raw)
training_Raw <- ptrain1[, -nzv]
testing_Raw <- testing_Raw[, -nzv]

mostlyNA <- sapply(ptrain1, function(x) mean(is.na(x))) > 0.95
training_Raw <- ptrain1[, mostlyNA==F]
testing_Raw <- testing_Raw[, mostlyNA==F]
```

```
training_Raw <- training_Raw[, -(1:5)]
testing_Raw <- testing_Raw[, -(1:5)]
```

Cross Validation

Refit model using full training set

```
fitControl <- trainControl(method="cv", number=3, verboseIter=F)
fit <- train(classe ~ ., data=training_Raw, method="rf", trControl=fitC
ontrol)
```

Testing or Predict 20 different test cases

```
Submission <- predict(fit, newdata=testing_Raw)
Submission
```

[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

2- Decision Tree Model

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(MASS)
library(caret)
```

Cross Validation

subSamples <- createDataPartition(y=training_Raw$classe, p=0.70, list=FALSE) subTraining <- training_Raw[subSamples,] subTesting <- training_Raw[-subSamples,]

FitDT <- rpart(classe ~ ., data=subTraining, method="class") fancyRpartPlot(FitDT)

predictTreeMod1 <- predict(FitDT, subTesting, type = "class") rpart.plot(modFitDT, main="Classification Tree", extra=102, under=TRUE, faclen=0) CMDT <- confusionMatrix(predictTreeMod1, subTesting$classe) CMDT ```

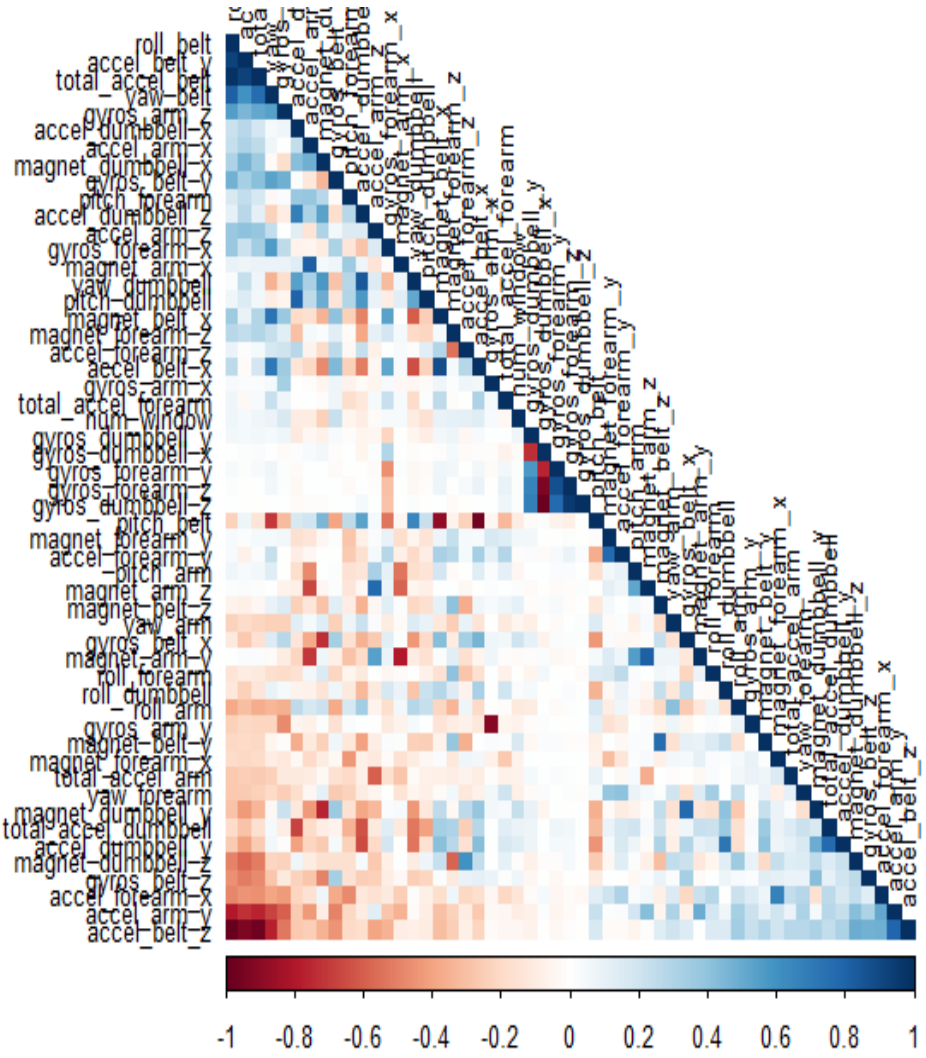
Result for DT Model:

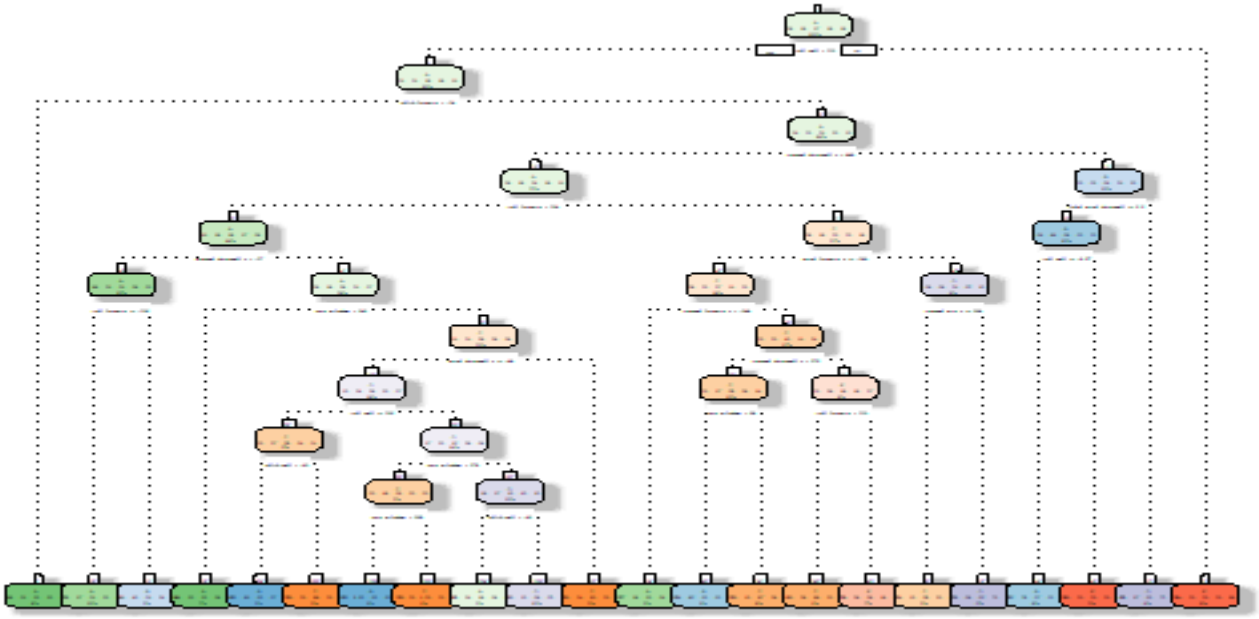
DT model has low accuracy (0.6967) and out-of-sample-error is about (0.3).

General Conclusion

The confusion matrices showed that the Random Forest algorithm accuracy (0.998) was better than decision tree accuracy (0.6967). ##Expected out-of-sample error The expected out-of-sample error is calculated as 1 - accuracy for predictions made against the cross-validation set. SO, out-of-sample-error of Decision Tree Model is about (0.3), while for Random Forest is (0.002).

Random Forest model was selected as the best model for prediction





Rattle 2020-سبتمبر-٠٨ ١٧:٣٨:٠٣ drall

Classification Tree