’Machine Learning Project

Abduallah Gamal

8/25/2019

Inroduction : 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, your goal will be to 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. Data Loading:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(caret)
```

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

```r
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
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(rpart)
library(rpart.plot)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
data.train<- read.csv("C:/Users/Abduallah Gamal/Downloads/pml-training.csv", n
a.strings = c("NA", "#DIV/0!", ""))

data.test<- read.csv("C:/Users/Abduallah Gamal/Downloads/pml-testing.csv", na.s
trings = c("NA", "#DIV/0!", ""))
```

Data Understandaing:

```r
dim(data.train)
```

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

Data Transformation : Convert date and add new variable (Day)

```r
data.train$cvtd_timestamp<- as.Date(data.train$cvtd_timestamp, format = "%m/%
d/%Y %H:%M")
data.train$Day<-factor(weekdays(data.train$cvtd_timestamp)) #Add day variable
```

Exploratory Data Analysis

```
table(data.train$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
prop.table(table(data.train$classe))
```

```
##
##         A         B         C         D         E
## 0.2843747 0.1935073 0.1743961 0.1638977 0.1838243
```

```
prop.table(table(data.train$user_name))
```

```
##
##    adelmo  carlitos   charles    eurico    jeremy     pedro
## 0.1983488 0.1585975 0.1802059 0.1564570 0.1733768 0.1330140
```

```
prop.table(table(data.train$user_name,data.train$classe),1)
```

```
##
##                      A         B         C         D         E
##   adelmo   0.2993320 0.1993834 0.1927030 0.1323227 0.1762590
##   carlitos 0.2679949 0.2217224 0.1584190 0.1561697 0.1956941
##   charles  0.2542421 0.2106900 0.1524321 0.1815611 0.2010747
##   eurico   0.2817590 0.1928339 0.1592834 0.1895765 0.1765472
##   jeremy   0.3459730 0.1437390 0.1916520 0.1534392 0.1651969
##   pedro    0.2452107 0.1934866 0.1911877 0.1796935 0.1904215
```

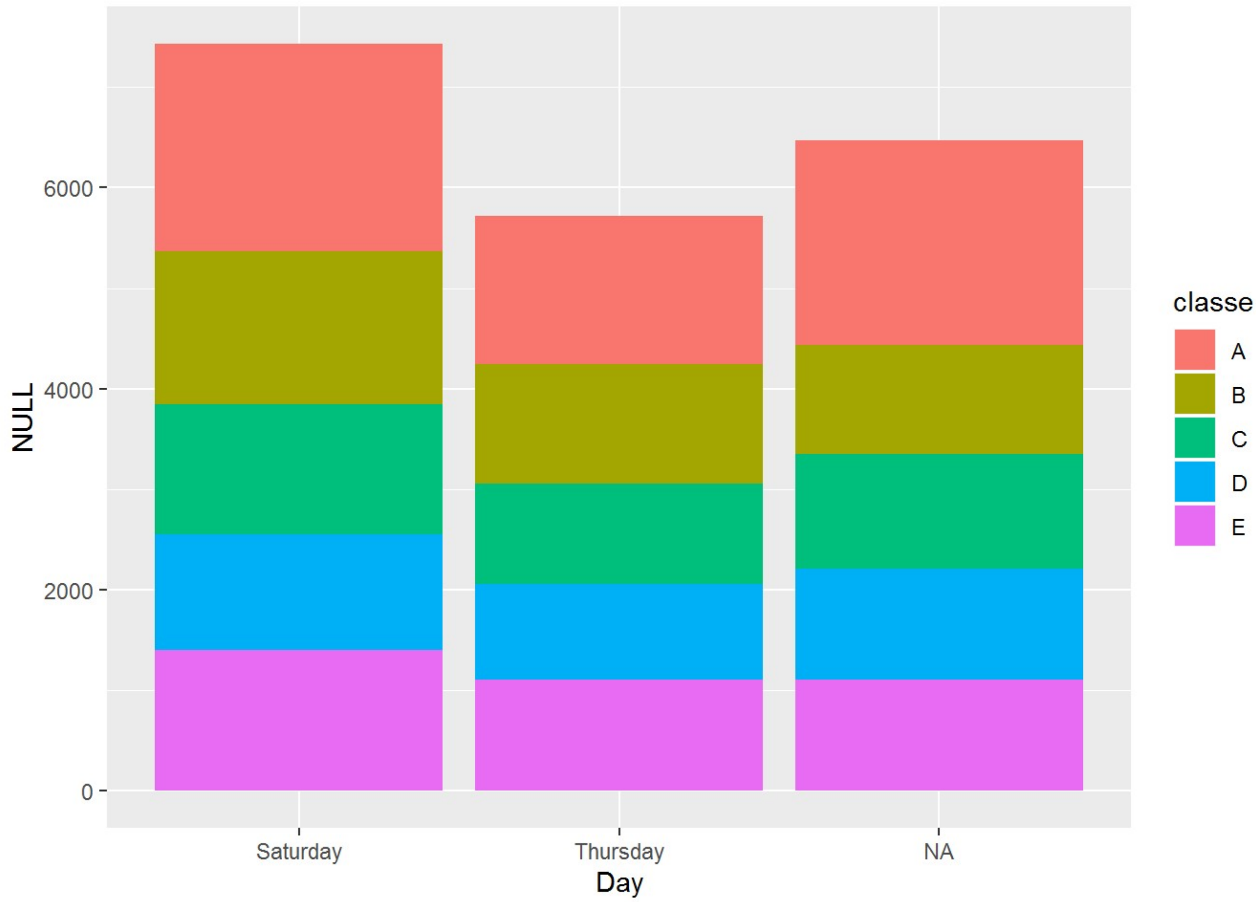
```
prop.table(table(data.train$user_name,data.train$classe),2)
```

```
##
##                      A         B         C         D         E
##   adelmo   0.2087814 0.2043719 0.2191701 0.1601368 0.1901857
##   carlitos 0.1494624 0.1817224 0.1440678 0.1511194 0.1688384
##   charles  0.1611111 0.1962075 0.1575102 0.1996269 0.1971167
##   eurico   0.1550179 0.1559126 0.1428989 0.1809701 0.1502634
##   jeremy   0.2109319 0.1287859 0.1905319 0.1623134 0.1558082
##   pedro    0.1146953 0.1329997 0.1458212 0.1458333 0.1377876
```

```
prop.table(table(data.train$classe, data.train$Day),1)
```

```
##
##      Saturday  Thursday
##    A 0.5833804 0.4166196
##    B 0.5600147 0.4399853
##    C 0.5651030 0.4348970
##    D 0.5478220 0.4521780
##    E 0.5581302 0.4418698
```

```
qplot(x=Day, fill=classe, data = data.train)
```



Key Insights from Exploratory Data Analysis:

```
### 1.Class-A activity is the most frequently used activity (28.5%) and is most
frequently used by user-Jeremy
### 2.Adelmo is the most frequent user of across acitivities (20%) but he uses
Class "C" activity most frequently.
### 3.Majority of the actitivies happened during Saturday's and Classes A and
B are the most frequently used activites.
```

Data Cleaning:

```
#### Remove columns with NA missing values
data.train <- data.train[, colSums(is.na(data.train)) == 0]
data.test <- data.test[, colSums(is.na(data.test)) == 0]

#### Remove columns that are not relevant to accelerometer measurements.
classe<- data.train$classe
trainRemove<- grepl("^X|timestamp|window", names(data.train))
data.train<- data.train[, !trainRemove]
trainCleaned<- data.train[, sapply(data.train, is.numeric)]
trainCleaned$classe<- classe
testRemove<- grepl("^X|timestamp|window", names(data.test))
data.test<- data.test[, !testRemove]
testCleaned<- data.test[, sapply(data.test, is.numeric)]
```

Now, the cleaned data contains 19622 observations and 53 variables for both train and test datasets
Create Train and Test data sets:

```
set.seed(22519)
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

Data Modelling:

```
#####Indetifying significant variables:
##### We will fit a predictive model using Random Forest algorithm as it gives
important variables and removes multicollinearity  and outliers. We will also u
se 5-fold cross validation when applying the algorithm.

controlRf <- trainControl(method="cv", 5)
rfmod<- train(classe ~., data=trainData, method="rf", trControl=controlRf, impo
rtance=TRUE, ntree=100)
rfmod
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10988, 10989, 10989, 10991, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9901007  0.9874765
##   27    0.9914834  0.9892262
##   52    0.9830388  0.9785437
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Accuacy of the model on Validation data set:

```
predictRfmod<- predict(rfmod, testData)
confusionMatrix(testData$classe, predictRfmod)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    1    3    0    0
##          B    6 1130    3    0    0
##          C    0    4 1020    2    0
##          D    0    0   11  952    1
##          E    0    0    5    2 1075
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9911, 0.9954)
##     No Information Rate : 0.2848
##     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            0.9964   0.9956   0.9789   0.9958   0.9991
## Specificity            0.9990   0.9981   0.9988   0.9976   0.9985
## Pos Pred Value         0.9976   0.9921   0.9942   0.9876   0.9935
## Neg Pred Value         0.9986   0.9989   0.9955   0.9992   0.9998
## Prevalence             0.2848   0.1929   0.1771   0.1624   0.1828
## Detection Rate         0.2838   0.1920   0.1733   0.1618   0.1827
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9977   0.9968   0.9888   0.9967   0.9988
```

```
accuracy <- postResample(predictRfmod, testData$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9935429 0.9918323
```

```
Error <- 1 - as.numeric(confusionMatrix(testData$classe, predictRfmod)$overall
[1])
Error
```

```
## [1] 0.006457094
```

So, the estimated accuracy of the model is 99.32% and the estimated out-of-sample error is 0.68%.

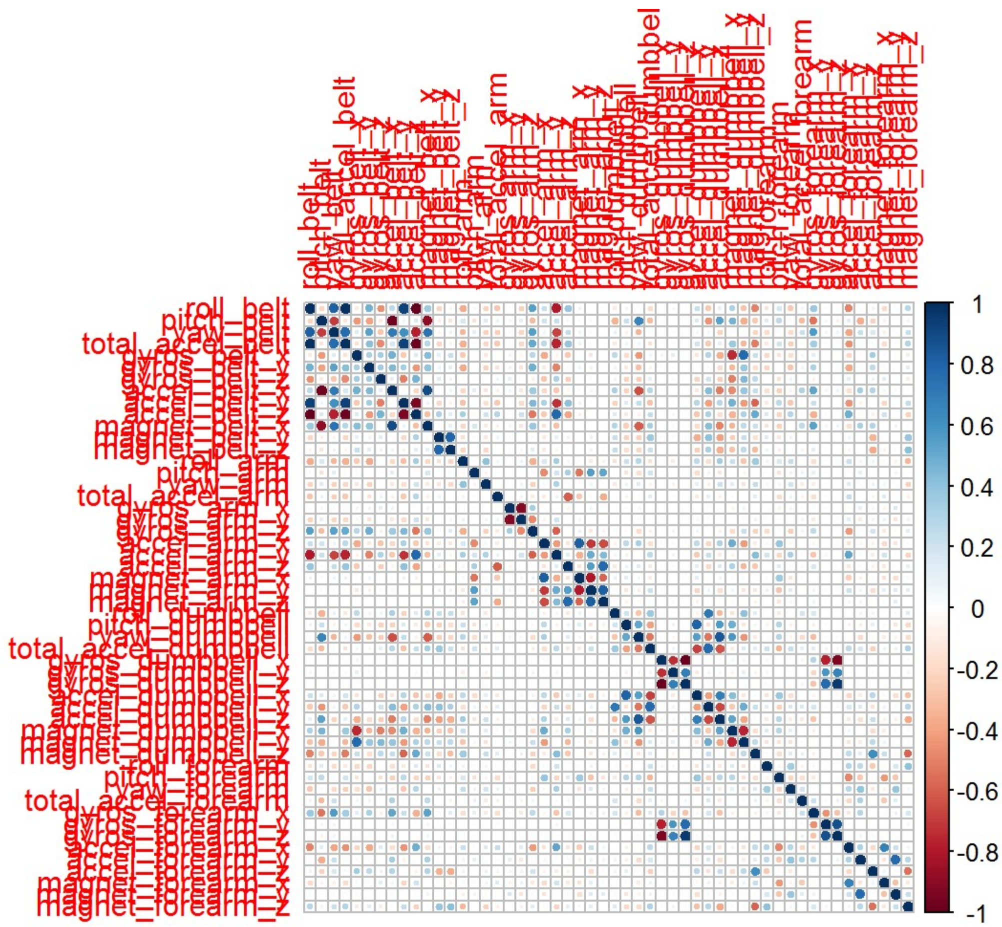
Predicting on Test Data Set

```
result <- predict(rfmod, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [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
```

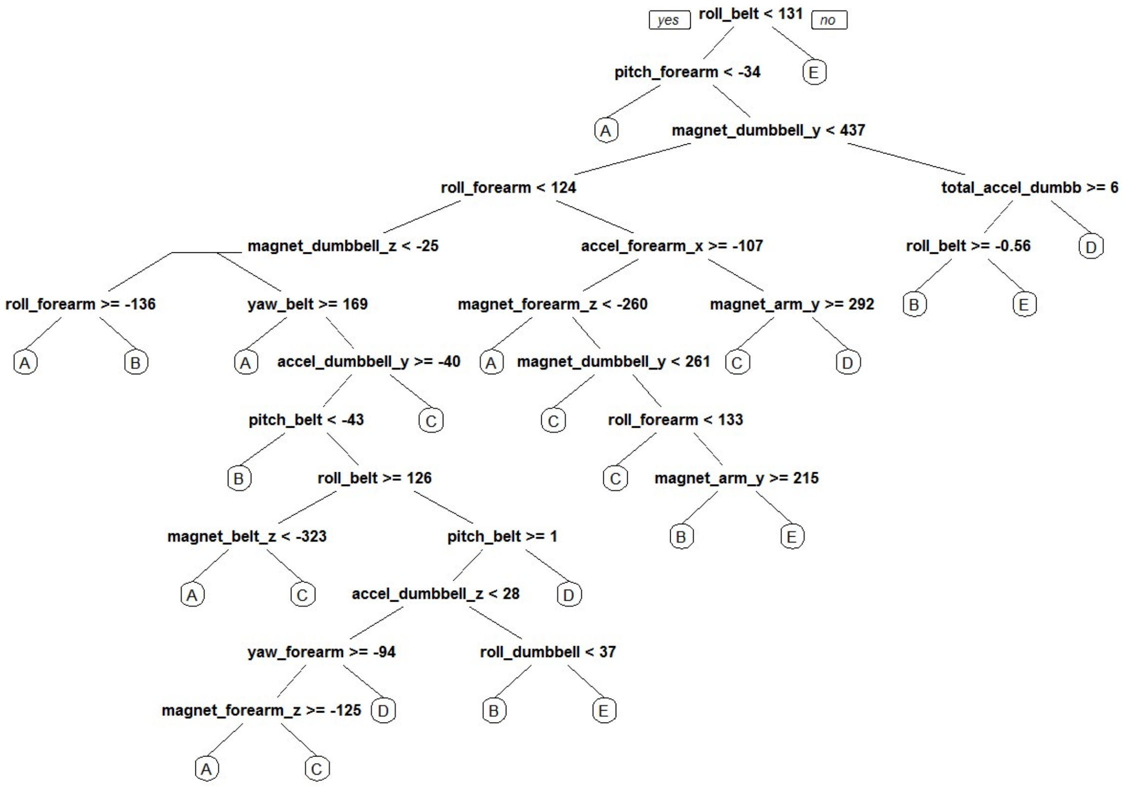
Appendix Correlation Matrix

```
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, method="circle")
```



Tree Visualization

```
rtree<- rpart(classe ~ ., data=trainData, method="class")
prp(rtree)
```



yes
roll_belt < 131
no
pitch_forearm < -34
E
A
magnet_dumbbell_y < 437
roll_forearm < 124
total_accel_dumbb >= 6
magnet_dumbbell_z < -25
accel_forearm_x >= -107
roll_belt >= -0.56
D
roll_forearm >= -136
yaw_belt >= 169
magnet_forearm_z < -260
magnet_arm_y >= 292
B
E
A
B
A
accel_dumbbell_y >= -40
A
magnet_dumbbell_y < 261
C
D
pitch_belt < -43
C
C
roll_forearm < 133
B
roll_belt >= 126
C
magnet_arm_y >= 215
magnet_belt_z < -323
pitch_belt >= 1
B
E
A
C
accel_dumbbell_z < 28
D
yaw_forearm >= -94
roll_dumbbell < 37
magnet_forearm_z >= -125
D
B
E
A
C