# Prediction Assignment Writeup

AP

9/14/2020

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. More information is available from the website here: Link (see the section on the Weight Lifting Exercise Dataset).

## Get and clean data

```
library(caret);library(kernlab);set.seed(1111);
```

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

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
testing = read.csv("pml-testing.csv", header = TRUE,na.strings=c("NA","#DIV/0!",""))
training = read.csv("pml-training.csv", header = TRUE,na.strings=c("NA","#DIV/0!",""))
```

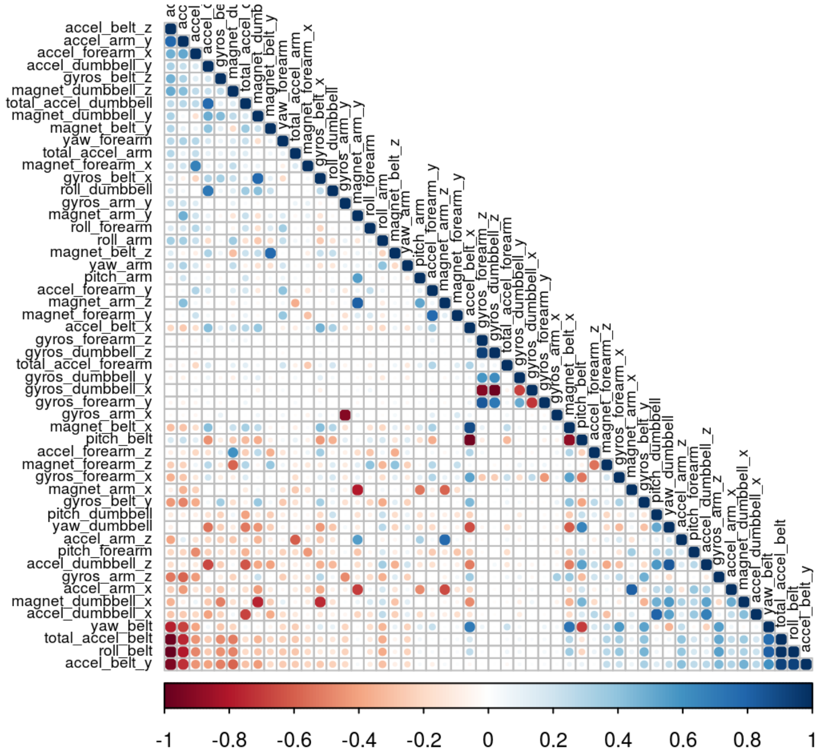
Remove variables with near zero variance and non predict variables

```
training <-training[,colSums(is.na(training)) == 0]
testing  <-testing[,colSums(is.na(testing)) == 0]
training <-training[,-c(1:7)]
testing  <-testing[,-c(1:7)]
```

```
# sapply(training, class)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corr_matrix <- cor(training[1:52])
corrplot(corr_matrix, order = "FPC", method = "circle", type = "lower",
         tl.cex = 0.6, tl.col = rgb(0, 0, 0))
```




No near Zero variance parameters

```
nearZeroVar(training,saveMetric=TRUE)
```

```
# The dimension of the two input databases
# head(training)
dim(training)
```

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

```
dim(testing)
```

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

The dataframes are of dimensions: * training - data frame with 160 observations on 19622 variables. * test - data frame with 160 observations on 20 variables.

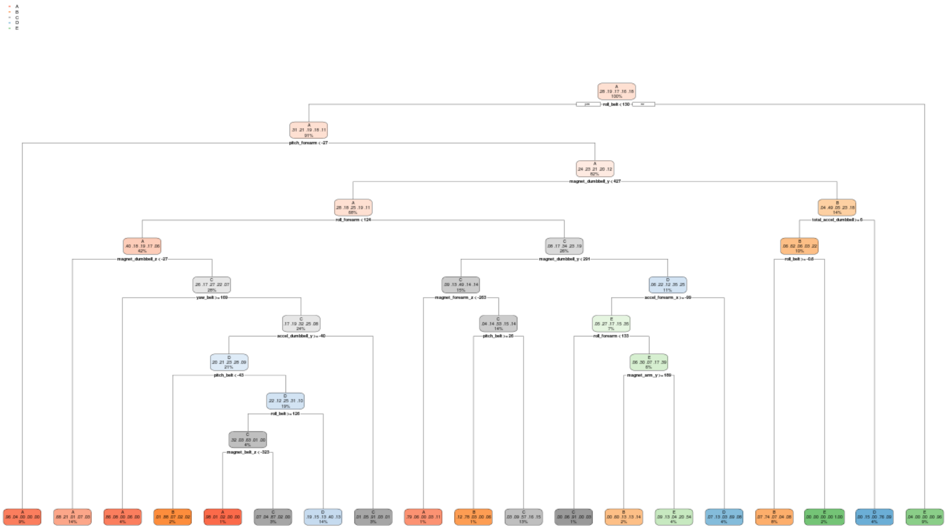
# Cross validation

Split the data into two sequences 75% for training and 25% for testing

```
set.seed(345)
trainingClasse <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
trainingA <- training[trainingClasse,]
testingA <- training[-trainingClasse,]
```

# METHOD 1 - Decision Tree

```
library(rpart); library(RColorBrewer);library(rpart.plot);
fit <- rpart(classe ~ ., data=trainingA, method="class")
rpart.plot(fit)
```



Use model to predict class in testing set

```
predictRpart <- predict(fit, testingA, type = "class")
confusionMatrix(predictRpart, testingA$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1147  199   12   78   26
##          B   30  476   52   35   68
##          C   39   83  668  132   88
##          D  146  165  116  525  131
##          E   33   26    7   34  588
##
## Overall Statistics
##
##                Accuracy : 0.6941
##                  95% CI : (0.681, 0.707)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6134
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8222  0.50158   0.7813   0.6530   0.6526
## Specificity            0.9102  0.95322   0.9155   0.8639   0.9750
## Pos Pred Value         0.7845  0.72012   0.6614   0.4848   0.8547
## Neg Pred Value         0.9279  0.88852   0.9520   0.9270   0.9258
## Prevalence             0.2845  0.19352   0.1743   0.1639   0.1837
## Detection Rate         0.2339  0.09706   0.1362   0.1071   0.1199
## Detection Prevalence   0.2981  0.13479   0.2060   0.2208   0.1403
## Balanced Accuracy      0.8662  0.72740   0.8484   0.7584   0.8138
```

Success Percentage on the test sequence

```
sum(predictRpart == testingA$classe)/length(predictRpart)*100
```

```
## [1] 69.41272
```

# METHOD 2 - RANDOM FOREST

```
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
```

```
fitRf <- randomForest(classe ~ ., data=trainingA, method="class")
```

Use model to predict class in testing set

```
predictRandFors <- predict(fitRf, testingA, type = "class")
confusionMatrix(predictRandFors, testingA$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    1    0    0    0
##          B    0  944    4    0    0
##          C    0    4  851    8    0
##          D    0    0    0  796    1
##          E    1    0    0    0  900
##
## Overall Statistics
##
##                Accuracy : 0.9961
##                  95% CI : (0.994, 0.9977)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9951
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9947   0.9953   0.9900   0.9989
## Specificity            0.9997   0.9990   0.9970   0.9998   0.9998
## Pos Pred Value         0.9993   0.9958   0.9861   0.9987   0.9989
## Neg Pred Value         0.9997   0.9987   0.9990   0.9981   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1925   0.1735   0.1623   0.1835
## Detection Prevalence   0.2845   0.1933   0.1760   0.1625   0.1837
## Balanced Accuracy      0.9995   0.9969   0.9962   0.9949   0.9993
```

Success Percentage on the test sequence

```
sum(predictRandFors == testingA$classe)/length(predictRandFors)*100
```

```
## [1] 99.61256
```

# METHOD 3 - Linear Discriminant Analysis

LDA is a classification method that finds a linear combination of data attributes that best separate the data into classes.

```r
library(MASS)
lm1 <- lda(classe ~ . , data=trainingA)
```

Use model to predict class in testing set

```r
pred <- predict(lm1, testingA[,1:52])$class
confusionMatrix(pred, testingA$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1144  136   86   48   30
##          B   35  607   70   29  154
##          C  124  127  575  107   91
##          D   86   32   96  589   85
##          E    6   47   28   31  541
## 
## Overall Statistics
## 
##                Accuracy : 0.7047
##                  95% CI : (0.6917, 0.7175)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.6264
## 
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8201   0.6396   0.6725   0.7326   0.6004
## Specificity            0.9145   0.9272   0.8891   0.9271   0.9720
## Pos Pred Value         0.7922   0.6782   0.5615   0.6633   0.8285
## Neg Pred Value         0.9275   0.9147   0.9278   0.9465   0.9153
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2333   0.1238   0.1173   0.1201   0.1103
## Detection Prevalence   0.2945   0.1825   0.2088   0.1811   0.1332
## Balanced Accuracy      0.8673   0.7834   0.7808   0.8298   0.7862
```

Success Percentage on the test sequence

```r
sum(pred == testingA$classe)/length(pred)*100
```

```
## [1] 70.47308
```

# METHOD 4 - Generalized Boosted Model (GBM)

```r
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```r
set.seed(345)
gbmCtrl <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
fit  <- train(classe ~ ., data = trainingA, method = "gbm", trControl = gbmCtrl, verbose = FALSE)
fit$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 51 had non-zero influence.
```

Use model to predict class in testing set

```
predictGbm <- predict(fit, testingA)
confusionMatrix(predictGbm, testingA$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1371   38    0    0    0
##          B   11  876   20    3    9
##          C    7   32  819   27    8
##          D    4    2   14  767   16
##          E    2    1    2    7  868
##
## Overall Statistics
##
##                Accuracy : 0.9586
##                  95% CI : (0.9526, 0.964)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9476
##
##  Mcnemar's Test P-Value : 5.049e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9828   0.9231   0.9579   0.9540   0.9634
## Specificity            0.9892   0.9891   0.9817   0.9912   0.9970
## Pos Pred Value         0.9730   0.9532   0.9171   0.9552   0.9864
## Neg Pred Value         0.9931   0.9817   0.9910   0.9910   0.9918
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2796   0.1786   0.1670   0.1564   0.1770
## Detection Prevalence   0.2873   0.1874   0.1821   0.1637   0.1794
## Balanced Accuracy      0.9860   0.9561   0.9698   0.9726   0.9802
```

Success Percentage on the test sequence

```
sum(predictGbm == testingA$classe)/length(predictGbm)*100
```

```
## [1] 95.86052
```

# Best Predictive Model to the Test Data

Summary of all predictions performance

- METHOD 1 - Decision Tree - 69.41272%
- METHOD 2 - RANDOM FOREST - 99.61256%
- METHOD 3 - Linear Discriminant Analysis - 70.47308%
- METHOD 4 - Generalized Boosted Model (GBM) - 95.86052%

Prediction of the test sequence according tho the random forest method.

```
predict(fitRf,testing)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```