Practical Machine Learning Assignment

About The Data

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: http://web.archive.org/web/20161224072740/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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Loading Packages

```
library(lattice)
library(ggplot2)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.3
```

```
library(kernlab)
```

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

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

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 4.1.3

## Loading required package: tibble

## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.1.3
```

```
## corrplot 0.92 loaded
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.1.3
```

```
## randomForest 4.7-1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
##
##     importance
```

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

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.1.3
```

```
## Loaded gbm 2.1.8
```

```
set.seed(2022)
```

Loading data into R

```
traincsv <- read.csv("pml-training.csv")
testcsv <- read.csv("pml-testing.csv")

dim(traincsv)
```

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

Removing unnecessary data

```
traincsv <- traincsv[, colMeans(is.na(traincsv)) < .9]
traincsv <- traincsv[, -c(1:7)]
```

N/A values

```
nvz <- nearZeroVar(traincsv)
traincsv <- traincsv[, -nvz]
dim(traincsv)
```

Removing near zero varience variables

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

Splitting training set into a Validation and Sub training set

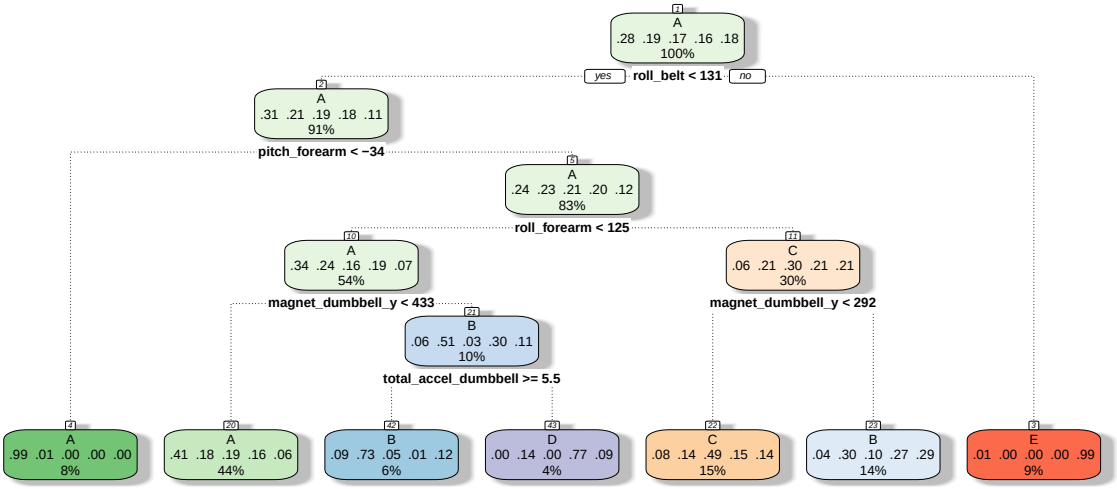
```
inTrain <- createDataPartition(y = traincsv$classe, p = 0.7, list = FALSE)
train <- traincsv[inTrain,]
valid <- traincsv[-inTrain,]
```

Creating and Testing the Models

```
control <- trainControl(method = "cv", number = 3, verboseIter =  FALSE)
```

Decision Tree

```
mod_trees <- train(classe ~ ., data = train, method = "rpart", trControl = control, tuneLength = 5)
fancyRpartPlot(mod_trees$finalModel)
```



Rattle 2022−Mar−30 21:49:15 R J Gawde

Model

Predition

```
pred_trees <- predict(mod_trees, valid)
cmtrees <- confusionMatrix(pred_trees, factor(valid$classe))
cmtrees
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1501  464  469  453  159
##          B   64  529  105  234  283
##          C  104  106  450  119  143
##          D    0   40    2  158   26
##          E    5    0    0    0  471
##
## Overall Statistics
##
##                Accuracy : 0.5283
##                  95% CI : (0.5154, 0.5411)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3829
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8967  0.46444  0.43860  0.16390  0.43530
## Specificity            0.6331  0.85546  0.90286  0.98618  0.99896
## Pos Pred Value         0.4928  0.43539  0.48807  0.69912  0.98950
## Neg Pred Value         0.9391  0.86938  0.88394  0.85757  0.88704
## Prevalence             0.2845  0.19354  0.17434  0.16381  0.18386
## Detection Rate         0.2551  0.08989  0.07647  0.02685  0.08003
## Detection Prevalence   0.5176  0.20646  0.15667  0.03840  0.08088
## Balanced Accuracy      0.7649  0.65995  0.67073  0.57504  0.71713
```

Random Forest

```
mod_rf <- train(classe ~., data = train , method = "rf", trControl = control, tuneLength = 5)

pred_rf <- predict(mod_rf, valid)
cmrf <- confusionMatrix(pred_rf, factor(valid$classe))
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    7    0    0    0
##          B    1 1129    8    0    0
##          C    1    3 1017   10    1
##          D    0    0    1  954    1
##          E    0    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9921, 0.9961)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9912   0.9912   0.9896   0.9982
## Specificity            0.9983   0.9981   0.9969   0.9996   1.0000
## Pos Pred Value         0.9958   0.9921   0.9855   0.9979   1.0000
## Neg Pred Value         0.9995   0.9979   0.9981   0.9980   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1918   0.1728   0.1621   0.1835
## Detection Prevalence   0.2853   0.1934   0.1754   0.1624   0.1835
## Balanced Accuracy      0.9986   0.9947   0.9941   0.9946   0.9991
```

Gradient Boosted Trees

```
mod_gbm <- train(classe ~ ., data = train, method = "gbm", trControl = control, tuneLength = 5, verbose

pred_gbm <- predict(mod_gbm, valid)
cmgbm <- confusionMatrix(pred_gbm, factor(valid$classe))
cmgbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1667    9    0    1    0
##          B    4 1123   11    0    0
##          C    3    7 1009   10    1
##          D    0    0    6  953    2
##          E    0    0    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.9908
##                  95% CI : (0.988, 0.9931)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9884
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9860   0.9834   0.9886   0.9972
## Specificity            0.9976   0.9968   0.9957   0.9984   1.0000
## Pos Pred Value         0.9940   0.9868   0.9796   0.9917   1.0000
## Neg Pred Value         0.9983   0.9966   0.9965   0.9978   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2833   0.1908   0.1715   0.1619   0.1833
## Detection Prevalence   0.2850   0.1934   0.1750   0.1633   0.1833
## Balanced Accuracy      0.9967   0.9914   0.9896   0.9935   0.9986
```

Support Vector machine

```
mod_svm <- train(classe ~ ., data = train, method = "svmLinear", trContorl = control, tuneLength = 5, v

pred_svm <- predict(mod_svm, valid)
cmsvm <- confusionMatrix(pred_svm, factor(valid$classe))
cmsvm
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1537  141   92   60   64
##          B   35  838   99   37  130
##          C   48   59  773  115   78
##          D   48   26   29  705   58
##          E    6   75   33   47  752
##
## Overall Statistics
##
##                Accuracy : 0.7825
##                  95% CI : (0.7717, 0.793)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7235
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9182   0.7357   0.7534   0.7313   0.6950
## Specificity            0.9152   0.9366   0.9383   0.9673   0.9665
## Pos Pred Value         0.8115   0.7357   0.7204   0.8141   0.8237
## Neg Pred Value         0.9657   0.9366   0.9474   0.9484   0.9336
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2612   0.1424   0.1314   0.1198   0.1278
## Detection Prevalence   0.3218   0.1935   0.1823   0.1472   0.1551
## Balanced Accuracy      0.9167   0.8362   0.8458   0.8493   0.8307
```

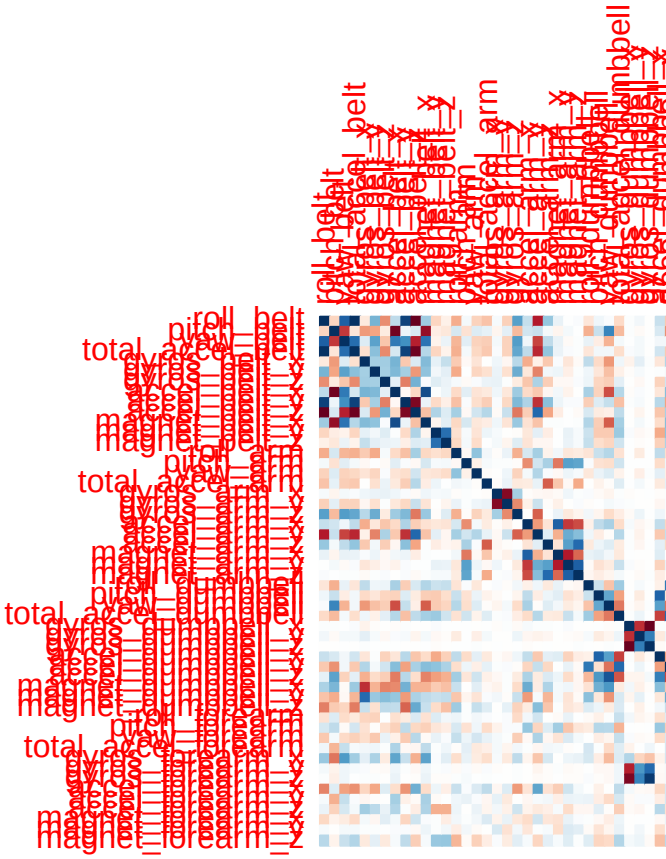
Predictions on Test Set

```
pred <- predict(mod_rf, testcsv)
print(pred)
```

```
##  [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
```

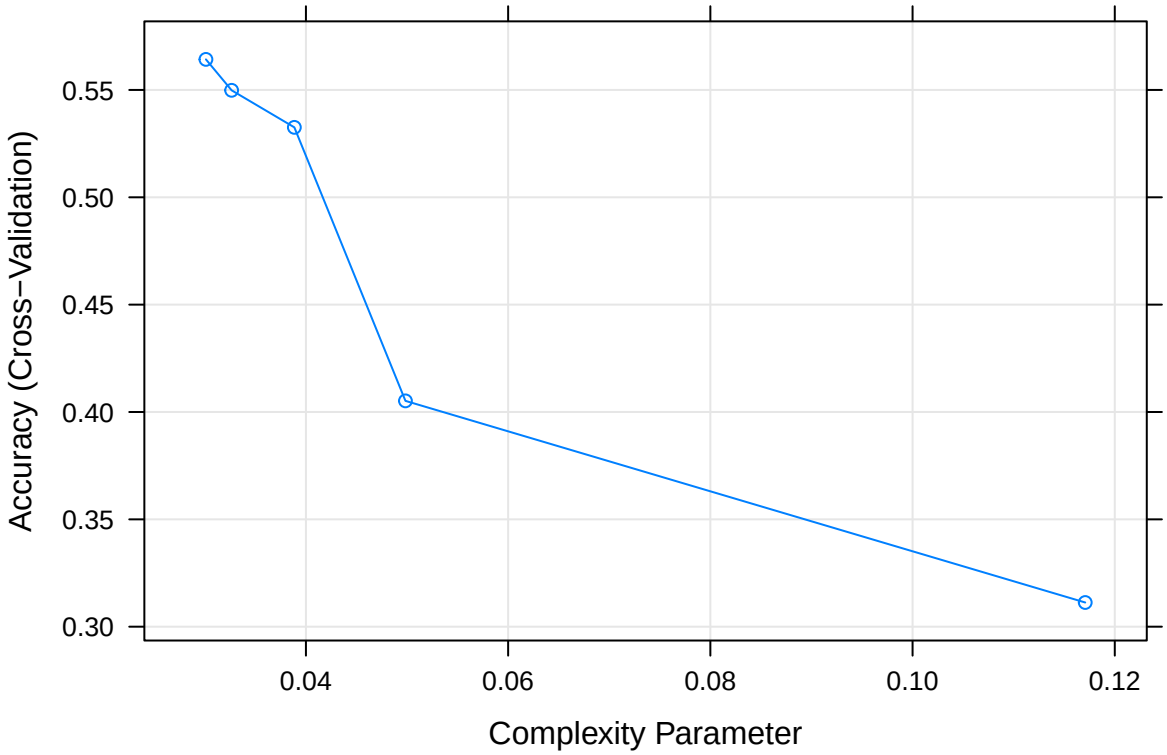
Plot

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



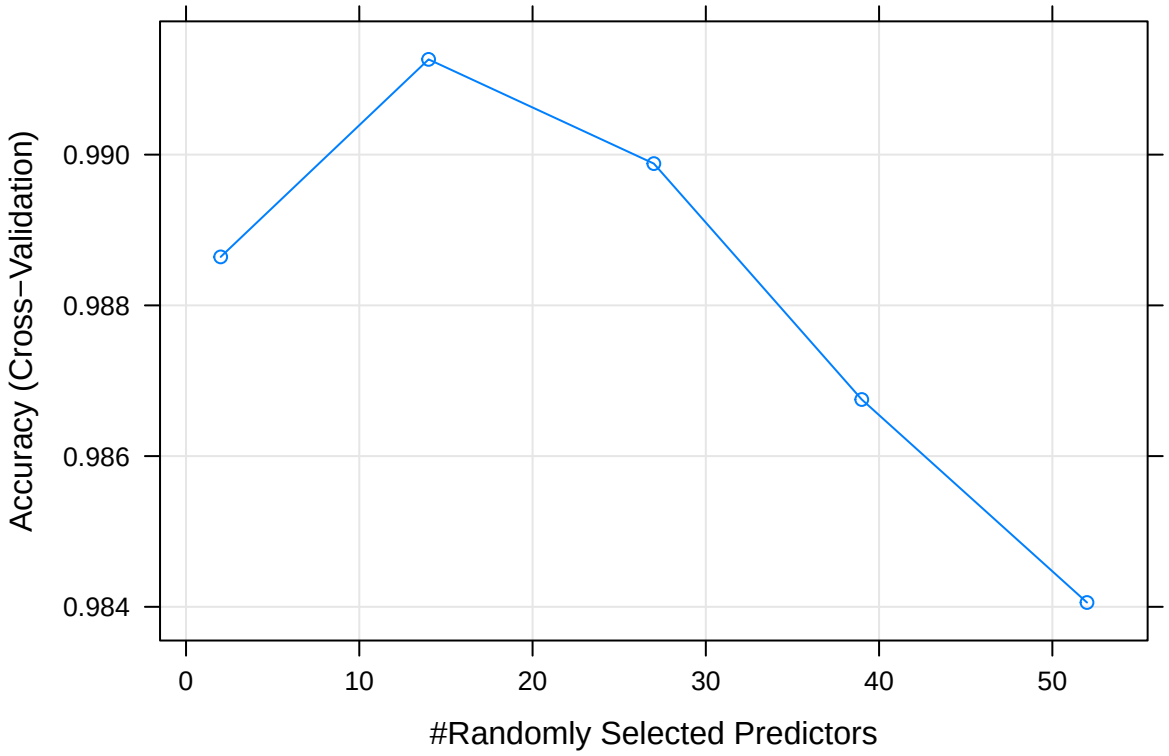
Correlation matrix of Variable in Training set

```
plot(mod_trees)
```

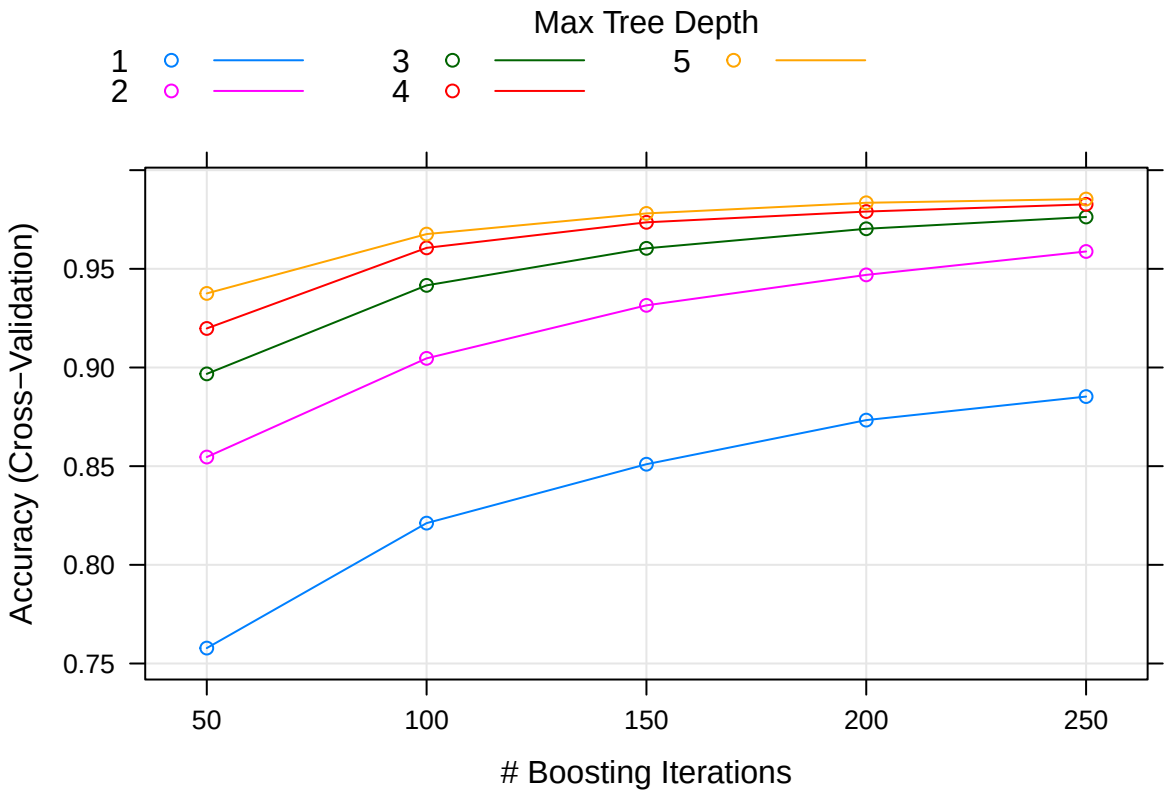


Plotting Models

```
plot(mod_rf)
```


Accuracy (Cross−Validation)
0.990
0.988
0.986
0.984
0
10
20
30
40
50
#Randomly Selected Predictors

```
plot(mod_gbm)
```


Max Tree Depth
1
2
3
4
5
Accuracy (Cross−Validation)
0.75
0.80
0.85
0.90
0.95
50
100
150
200
250
Boosting Iterations