Coursera Machine Learning Project

Louis

11/15/2020

I loaded the necessary libraries to do the machine learning.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.3
```

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

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

```
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(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.0.3
```

```
## Loading required package: rpart
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.0.3
```

```
## corrplot 0.84 loaded
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.3

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

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

```r
library(latexpdf)
```

The test and training data was loaded and I created an data partition where I then made training and test validation data sets based on the data partition.

```r
train_data <- read.csv("pml-training.csv")
test_data <- read.csv("pml-testing.csv")
intrain <- createDataPartition(train_data$classe, p=0.7, list = FALSE)
train_valid <- train_data[intrain,]
test_valid <- train_data[-intrain,]
```

Get rid of variables with zero variables then removed NA data to make the training as efficient as possible.

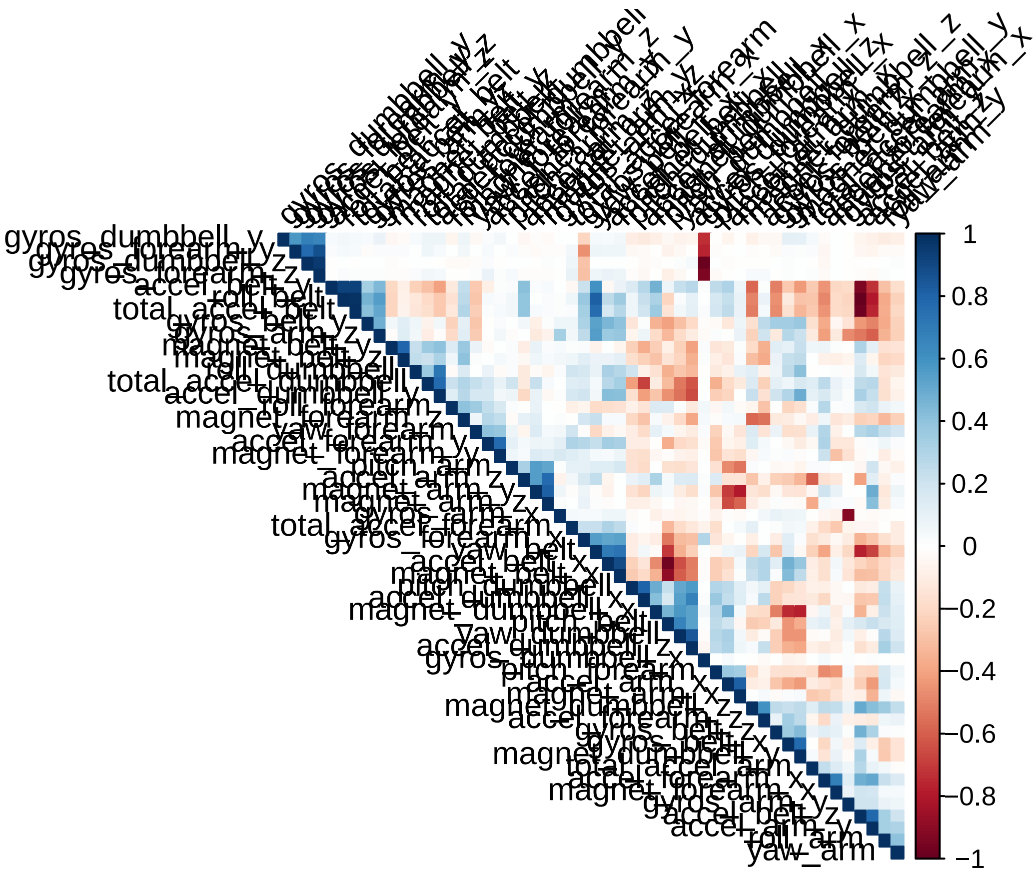
```r
near_zero <- nearZeroVar(train_valid)
train_valid <- train_valid[,-near_zero]
test_valid <- test_valid[, -near_zero]

all_na <- sapply(train_valid,function(l) mean(is.na(l))) > 0.95
train_valid <- train_valid[, all_na == FALSE]
test_valid <- test_valid[, all_na == FALSE]
```

#Remove uncessary variables

```
train_valid <- train_valid[, -c(1:6)]
test_valid <- test_valid[, -c(1:6)]
```

Find what data is correlated with one another and used a cutoff of 0.7 to determine the best correlation

```
corr_train <- cor(train_valid[, -53], use = "pairwise.complete.obs")
corr_high <- findCorrelation(corr_train, cutoff = 0.7)
#corrplot doesn't help much
corrplot(corr_train,order = "hclust" , method = "color" ,type = "upper",tl.col="black", tl.srt=45)
```



Train the necessary prediction models and compare the accuracy of the seperate prediction models to find the best model to use.

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

```
#Prediction models
#Random Forest
set.seed(13)
fit.rf <- train(classe~., data=train_valid, method="rf", metric=metric, trControl=control)
#Classification Tree
set.seed(13)
fit.ct <- train(classe~., data=train_valid, method="rpart", trControl=control)
#Linear Algorithms
set.seed(13)
fit.la <- train(classe~., data=train_valid, method="lda", metric = metric, trControl=control)

#Rpart
set.seed(13)
fit.cart <- train(classe~., data=train_valid, method="rpart", metric=metric, trControl=control)

#Gradient Boosting method
set.seed(13)
fit.gbm <- train(classe~., data=train_valid, method="gbm", metric=metric, trControl=control)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1255
##      2        1.5243             nan     0.1000    0.0833
##      3        1.4669             nan     0.1000    0.0687
##      4        1.4209             nan     0.1000    0.0553
##      5        1.3854             nan     0.1000    0.0442
##      6        1.3564             nan     0.1000    0.0444
##      7        1.3273             nan     0.1000    0.0410
##      8        1.3021             nan     0.1000    0.0336
##      9        1.2804             nan     0.1000    0.0315
##     10        1.2586             nan     0.1000    0.0296
##     20        1.1049             nan     0.1000    0.0156
##     40        0.9300             nan     0.1000    0.0084
##     60        0.8232             nan     0.1000    0.0062
##     80        0.7422             nan     0.1000    0.0044
##    100        0.6787             nan     0.1000    0.0028
##    120        0.6269             nan     0.1000    0.0026
##    140        0.5824             nan     0.1000    0.0014
##    150        0.5625             nan     0.1000    0.0023
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1904
##      2        1.4885             nan     0.1000    0.1267
##      3        1.4050             nan     0.1000    0.1025
##      4        1.3391             nan     0.1000    0.0844
##      5        1.2842             nan     0.1000    0.0682
##      6        1.2408             nan     0.1000    0.0606
##      7        1.2020             nan     0.1000    0.0573
##      8        1.1639             nan     0.1000    0.0513
##      9        1.1306             nan     0.1000    0.0501
##     10        1.0987             nan     0.1000    0.0453
##     20        0.8917             nan     0.1000    0.0203
##     40        0.6787             nan     0.1000    0.0086
##     60        0.5513             nan     0.1000    0.0064
```

```
##     80        0.4617             nan     0.1000    0.0048
##    100        0.3963             nan     0.1000    0.0025
##    120        0.3439             nan     0.1000    0.0023
##    140        0.3052             nan     0.1000    0.0017
##    150        0.2864             nan     0.1000    0.0021
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2304
##      2        1.4605             nan     0.1000    0.1557
##      3        1.3612             nan     0.1000    0.1258
##      4        1.2812             nan     0.1000    0.1045
##      5        1.2161             nan     0.1000    0.0881
##      6        1.1589             nan     0.1000    0.0795
##      7        1.1091             nan     0.1000    0.0628
##      8        1.0682             nan     0.1000    0.0618
##      9        1.0285             nan     0.1000    0.0648
##     10        0.9876             nan     0.1000    0.0513
##     20        0.7514             nan     0.1000    0.0210
##     40        0.5300             nan     0.1000    0.0135
##     60        0.4004             nan     0.1000    0.0070
##     80        0.3179             nan     0.1000    0.0036
##    100        0.2603             nan     0.1000    0.0028
##    120        0.2169             nan     0.1000    0.0021
##    140        0.1831             nan     0.1000    0.0017
##    150        0.1695             nan     0.1000    0.0014
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1254
##      2        1.5236             nan     0.1000    0.0830
##      3        1.4671             nan     0.1000    0.0669
##      4        1.4238             nan     0.1000    0.0489
##      5        1.3905             nan     0.1000    0.0514
##      6        1.3569             nan     0.1000    0.0474
##      7        1.3276             nan     0.1000    0.0411
##      8        1.3017             nan     0.1000    0.0349
##      9        1.2795             nan     0.1000    0.0339
##     10        1.2572             nan     0.1000    0.0309
##     20        1.0996             nan     0.1000    0.0146
##     40        0.9251             nan     0.1000    0.0081
##     60        0.8149             nan     0.1000    0.0053
##     80        0.7358             nan     0.1000    0.0059
##    100        0.6696             nan     0.1000    0.0034
##    120        0.6196             nan     0.1000    0.0023
##    140        0.5745             nan     0.1000    0.0025
##    150        0.5528             nan     0.1000    0.0030
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1807
##      2        1.4891             nan     0.1000    0.1247
##      3        1.4075             nan     0.1000    0.1071
##      4        1.3386             nan     0.1000    0.0872
##      5        1.2829             nan     0.1000    0.0760
##      6        1.2346             nan     0.1000    0.0573
##      7        1.1959             nan     0.1000    0.0620
```

```
##      8        1.1570             nan     0.1000    0.0561
##      9        1.1214             nan     0.1000    0.0446
##     10        1.0933             nan     0.1000    0.0541
##     20        0.8835             nan     0.1000    0.0257
##     40        0.6715             nan     0.1000    0.0126
##     60        0.5422             nan     0.1000    0.0068
##     80        0.4599             nan     0.1000    0.0059
##    100        0.3925             nan     0.1000    0.0034
##    120        0.3404             nan     0.1000    0.0035
##    140        0.2985             nan     0.1000    0.0012
##    150        0.2818             nan     0.1000    0.0022
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2381
##      2        1.4580             nan     0.1000    0.1548
##      3        1.3589             nan     0.1000    0.1262
##      4        1.2777             nan     0.1000    0.1069
##      5        1.2095             nan     0.1000    0.0953
##      6        1.1493             nan     0.1000    0.0737
##      7        1.1010             nan     0.1000    0.0686
##      8        1.0549             nan     0.1000    0.0631
##      9        1.0146             nan     0.1000    0.0557
##     10        0.9784             nan     0.1000    0.0544
##     20        0.7463             nan     0.1000    0.0289
##     40        0.5219             nan     0.1000    0.0122
##     60        0.3918             nan     0.1000    0.0053
##     80        0.3132             nan     0.1000    0.0040
##    100        0.2568             nan     0.1000    0.0031
##    120        0.2148             nan     0.1000    0.0019
##    140        0.1835             nan     0.1000    0.0015
##    150        0.1692             nan     0.1000    0.0009
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1297
##      2        1.5205             nan     0.1000    0.0837
##      3        1.4617             nan     0.1000    0.0679
##      4        1.4166             nan     0.1000    0.0532
##      5        1.3820             nan     0.1000    0.0445
##      6        1.3514             nan     0.1000    0.0470
##      7        1.3220             nan     0.1000    0.0415
##      8        1.2961             nan     0.1000    0.0299
##      9        1.2755             nan     0.1000    0.0340
##     10        1.2525             nan     0.1000    0.0301
##     20        1.0947             nan     0.1000    0.0154
##     40        0.9174             nan     0.1000    0.0083
##     60        0.8098             nan     0.1000    0.0059
##     80        0.7314             nan     0.1000    0.0044
##    100        0.6682             nan     0.1000    0.0023
##    120        0.6168             nan     0.1000    0.0025
##    140        0.5744             nan     0.1000    0.0022
##    150        0.5569             nan     0.1000    0.0016
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1854
```

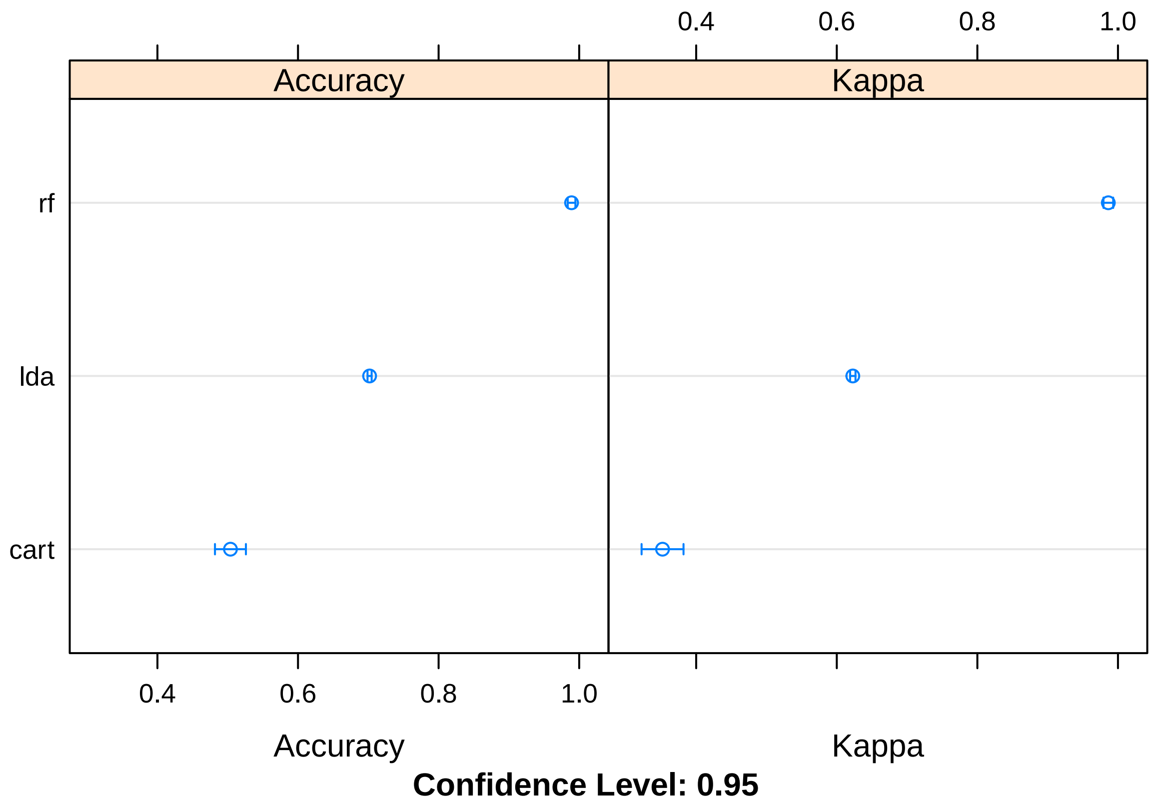
```
##      2        1.4867             nan     0.1000    0.1332
##      3        1.4002             nan     0.1000    0.1041
##      4        1.3338             nan     0.1000    0.0790
##      5        1.2822             nan     0.1000    0.0726
##      6        1.2366             nan     0.1000    0.0675
##      7        1.1921             nan     0.1000    0.0634
##      8        1.1517             nan     0.1000    0.0449
##      9        1.1215             nan     0.1000    0.0459
##     10        1.0930             nan     0.1000    0.0386
##     20        0.8874             nan     0.1000    0.0254
##     40        0.6783             nan     0.1000    0.0084
##     60        0.5489             nan     0.1000    0.0088
##     80        0.4603             nan     0.1000    0.0053
##    100        0.3927             nan     0.1000    0.0020
##    120        0.3419             nan     0.1000    0.0022
##    140        0.2999             nan     0.1000    0.0023
##    150        0.2815             nan     0.1000    0.0016
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2402
##      2        1.4537             nan     0.1000    0.1603
##      3        1.3495             nan     0.1000    0.1250
##      4        1.2698             nan     0.1000    0.1008
##      5        1.2046             nan     0.1000    0.0840
##      6        1.1500             nan     0.1000    0.0807
##      7        1.0989             nan     0.1000    0.0586
##      8        1.0600             nan     0.1000    0.0604
##      9        1.0223             nan     0.1000    0.0580
##     10        0.9859             nan     0.1000    0.0514
##     20        0.7474             nan     0.1000    0.0248
##     40        0.5277             nan     0.1000    0.0140
##     60        0.4025             nan     0.1000    0.0058
##     80        0.3242             nan     0.1000    0.0041
##    100        0.2614             nan     0.1000    0.0031
##    120        0.2204             nan     0.1000    0.0018
##    140        0.1860             nan     0.1000    0.0018
##    150        0.1707             nan     0.1000    0.0010
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2338
##      2        1.4595             nan     0.1000    0.1600
##      3        1.3568             nan     0.1000    0.1215
##      4        1.2797             nan     0.1000    0.1062
##      5        1.2138             nan     0.1000    0.0977
##      6        1.1532             nan     0.1000    0.0786
##      7        1.1030             nan     0.1000    0.0706
##      8        1.0591             nan     0.1000    0.0522
##      9        1.0250             nan     0.1000    0.0641
##     10        0.9841             nan     0.1000    0.0557
##     20        0.7483             nan     0.1000    0.0210
##     40        0.5223             nan     0.1000    0.0135
##     60        0.4029             nan     0.1000    0.0083
##     80        0.3198             nan     0.1000    0.0051
##    100        0.2648             nan     0.1000    0.0021
```

```
##    120        0.2228             nan     0.1000    0.0016
##    140        0.1887             nan     0.1000    0.0015
##    150        0.1741             nan     0.1000    0.0014
```

```
results <- resamples(list(lda=fit.la, cart=fit.ct, rf=fit.rf))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, cart, rf
## Number of resamples: 3
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda  0.7005242 0.7013213 0.7021184 0.7018270 0.7024784 0.7028384    0
## cart 0.4936681 0.5012028 0.5087374 0.5038957 0.5090095 0.5092815    0
## rf   0.9871123 0.9879885 0.9888646 0.9891533 0.9901737 0.9914829    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda  0.6211090 0.6221066 0.6231041 0.6227853 0.6236235 0.6241429    0
## cart 0.3382773 0.3486582 0.3590391 0.3521455 0.3590796 0.3591201    0
## rf   0.9836938 0.9848030 0.9859123 0.9862776 0.9875696 0.9892268    0
```

```
dotplot(results)
```



We found that the random forest model is the most accurate according to the boxplot.

Use predict function to test the random forest model

```
end_result <- predict(fit.rf, newdata = test_data)
summary(end_result)
```

```
## A B C D E
## 7 8 1 1 3
```

```
end_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
```