Course_Project_PML

D.Dashinov

12/5/2020

#Data loading and processing

Packages

```
library(caret)
```

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

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

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

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
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:rattle':
##
##     importance
```

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

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

The data came in horrible shape including empty columns thousands of NAs, excel errors and redundant calculations such as min, max, skewness and kurtosis. All of these were remove prior to loading in R for both the train and valid data set

```
train <- read.csv('./pml-training.csv', header=T)
valid <- read.csv('./pml-testing.csv', header=T)
dim(train)
```

```
## [1] 19622    60
```

```
dim(valid)
```

```
## [1] 20 60
```

We remove the first 7 variables

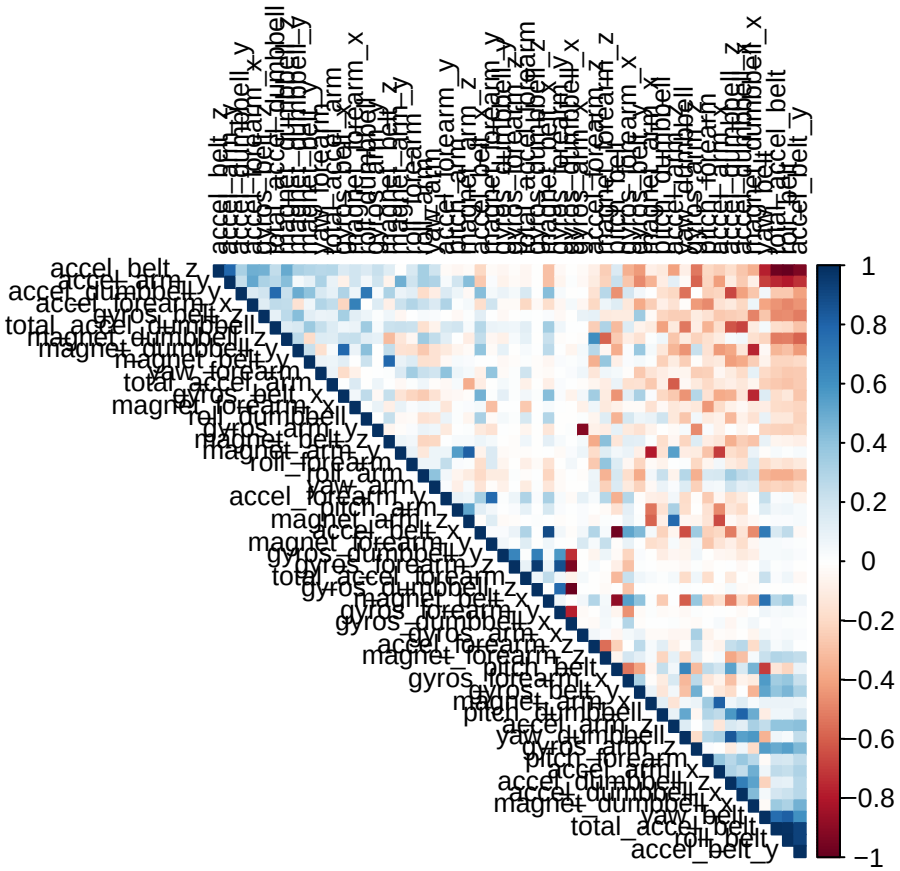
```
trainData <- train[, -c(1:7)]
validData <- valid[, -c(1:7)]
```

Subset train and test data

```
set.seed(1234)
inTrain <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
trainData <- trainData[inTrain, ]
testData <- trainData[-inTrain, ]
trainData$classe <- as.factor(trainData$classe)
testData$classe <- as.factor(testData$classe)
```

#Correlation of the variables

```
cor_mat <- cor(trainData[, -53])
corrplot(cor_mat, order = "FPC", method = "color", type = "upper",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



We use the findCorrelation function to search for highly correlated attributes with a cut off equal to 0.75

```
highlyCorrelated = findCorrelation(cor_mat, cutoff=0.75)
names(trainData)[highlyCorrelated]
```

```
##  [1] "accel_belt_z"      "roll_belt"         "accel_belt_y"
##  [4] "total_accel_belt"  "accel_dumbbell_z"  "accel_belt_x"
##  [7] "pitch_belt"        "magnet_dumbbell_x" "accel_dumbbell_y"
## [10] "magnet_dumbbell_y" "accel_dumbbell_x"  "accel_arm_x"
## [13] "accel_arm_z"       "magnet_arm_y"      "magnet_belt_z"
## [16] "accel_forearm_y"   "gyros_forearm_y"   "gyros_dumbbell_x"
## [19] "gyros_dumbbell_z"  "gyros_arm_x"
```

#Modeling Since the outcome is categorical we will try a classification tree, random forest and generalized boosted method ###Tree

```
set.seed(12345)
decisionTreeMod1 <- rpart(classe ~ ., data=trainData, method="class")
predictTreeMod1 <- predict(decisionTreeMod1, testData, type = "class")
cmtree <- confusionMatrix(predictTreeMod1, testData$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1067  105    9   24    9
##          B   40  502   59   63   77
##          C   28   90  611  116   86
```

```
##          D   11   49   41  423   41
##          E   19   41   18   46  548
##
## Overall Statistics
##
##                Accuracy : 0.7642
##                  95% CI : (0.751, 0.7771)
##     No Information Rate : 0.2826
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7015
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9159   0.6379   0.8279   0.6295   0.7201
## Specificity            0.9503   0.9284   0.9055   0.9589   0.9631
## Pos Pred Value         0.8789   0.6775   0.6563   0.7487   0.8155
## Neg Pred Value         0.9663   0.9157   0.9602   0.9300   0.9383
## Prevalence             0.2826   0.1909   0.1790   0.1630   0.1846
## Detection Rate         0.2588   0.1218   0.1482   0.1026   0.1329
## Detection Prevalence   0.2944   0.1797   0.2258   0.1370   0.1630
## Balanced Accuracy      0.9331   0.7831   0.8667   0.7942   0.8416
```

The Accuracy is 0.7415 of this model ###Random forest

```
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modRF1 <- train(classe ~ ., data=trainData, method="rf", trControl=controlRF)
modRF1$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: 0.7%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    3    0    0    1 0.001024066
## B   19 2634    5    0    0 0.009029345
## C    0   17 2369   10    0 0.011268781
## D    0    1   26 2224    1 0.012433393
## E    0    2    5    6 2512 0.005148515
```

```
predictRFMod <- predict(modRF1$finalModel, testData, type = "class")
cmRf <- confusionMatrix(predictRFMod, testData$classe)
cmRf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1165    0    0    0    0
##          B    0  787    0    0    0
##          C    0    0  738    0    0
##          D    0    0    0  672    0
##          E    0    0    0    0  761
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2826
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000    1.000    1.000   1.0000
## Specificity            1.0000   1.0000    1.000    1.000   1.0000
## Pos Pred Value         1.0000   1.0000    1.000    1.000   1.0000
## Neg Pred Value         1.0000   1.0000    1.000    1.000   1.0000
## Prevalence             0.2826   0.1909    0.179    0.163   0.1846
## Detection Rate         0.2826   0.1909    0.179    0.163   0.1846
## Detection Prevalence   0.2826   0.1909    0.179    0.163   0.1846
## Balanced Accuracy      1.0000   1.0000    1.000    1.000   1.0000
```

Obviously since the accuracy is 1 we have overfiting ###Boosting

```
set.seed(12345)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modGBM  <- train(classe ~ ., data=trainData, method = "gbm", trControl = controlGBM, verbose = FALSE)
modGBM$finalModel
```

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

Validate the GBM

```
predictGBM <- predict(modGBM, newdata=testData)
cmGBM <- confusionMatrix(predictGBM, testData$classe)
cmGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1155   20    0    0    1
##          B    9  754   17    5    6
##          C    1   12  713   16    3
##          D    0    1    6  647    8
##          E    0    0    2    4  743
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9731
##                  95% CI : (0.9677, 0.9778)
##     No Information Rate : 0.2826
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.966
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9914   0.9581   0.9661   0.9628   0.9763
## Specificity            0.9929   0.9889   0.9905   0.9957   0.9982
## Pos Pred Value         0.9821   0.9532   0.9570   0.9773   0.9920
## Neg Pred Value         0.9966   0.9901   0.9926   0.9928   0.9947
## Prevalence             0.2826   0.1909   0.1790   0.1630   0.1846
## Detection Rate         0.2801   0.1829   0.1729   0.1569   0.1802
## Detection Prevalence   0.2852   0.1919   0.1807   0.1606   0.1817
## Balanced Accuracy      0.9922   0.9735   0.9783   0.9792   0.9873
```

The accuracy is 0.9801

#Applying the best model to the validation data set (the GBM)

```
Results <- predict(modRF1, newdata=validData)
Results
```

```
##  [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
```