Prediction

21/10/2020

Background

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

Task

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

1. Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders.
2. You should also apply your machine learning algorithm to the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

```r
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.
```

```r
library(caret)
```

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

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

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

## corrplot 0.84 loaded
```

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

```
library(RColorBrewer)

set.seed(122333)
```

Data loading and cleaning

Now this is test to see if the dataset is downloaded in folder and if not it download it.

```
trainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainFile <- "pml-training.csv"
testFile  <- "pml-testing.csv"
if (!file.exists(trainFile)) {
  download.file(trainUrl, destfile = trainFile, method = "curl")
}
if (!file.exists(testFile)) {
  download.file(testUrl, destfile = testFile, method = "curl")
}


pml_training<-read.csv("pml-training.csv")
pml_testing<-read.csv("pml-testing.csv")

dim(pml_training)
```

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

```
dim(pml_testing)
```

```
## [1]  20 160
```

The training data set contains 19622 observations and 160 variables, while the testing data set contains 20 observations and 160 variables. The "classe" variable in the training set is the outcome to predict.

Clean the data

```
pml_training <- pml_training[, colSums(is.na(pml_training)) == 0]
pml_testing <- pml_testing[, colSums(is.na(pml_testing)) == 0]
```

Next, we get rid of some columns that do not contribute much to the accelerometer measurements.

```
classe <- pml_training$classe
pml_training_remove <- grepl("^X|timestamp|window", names(pml_training))
pml_training <- pml_training[, !pml_training_remove]
pml_training_cleaned <- pml_training[, sapply(pml_training, is.numeric)]
pml_training_cleaned$classe <- classe

pml_testing_remove <- grepl("^X|timestamp|window", names(pml_testing))
pml_testing <- pml_testing[, !pml_testing_remove]
pml_testing_cleaned <- pml_testing[, sapply(pml_testing, is.numeric)]
```

Now, the cleaned training data set contains 19622 observations and 53 variables, while the testing data set contains 20 observations and 53 variables. The "classe" variable is still in the cleaned training set.

Slice the data

Then, we can split the cleaned training set into a pure training data set (70%) and a validation data set (30%). We will use the validation data set to conduct cross validation in future steps.

```
inTrain <- createDataPartition(pml_training_cleaned$classe, p=0.70, list=FALSE)
trainData <- pml_training_cleaned[inTrain, ]
testData <- pml_training_cleaned[-inTrain, ]
```

Data Modeling

We fit a predictive model for activity recognition using Random Forest algorithm because it automatically selects important variables and is robust to correlated covariates & outliers in general. We will use 5-fold cross validation when applying the algorithm.

```r
controlRf <- trainControl(method="cv", 5)
modelRf <- train(classe ~ ., data=trainData, method="rf", trControl=controlRf, ntree=250)
modelRf
```

```
## 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: 10990, 10990, 10988, 10991, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9903180  0.9877517
##   27    0.9901718  0.9875666
##   52    0.9828190  0.9782636
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Then, we estimate the performance of the model on the validation data set.

```r
predictRf <- predict(modelRf, testData)
confusionMatrix(as.factor(testData$classe), predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    4 1135    0    0    0
##          C    0    5 1019    2    0
##          D    0    0   14  949    1
##          E    0    0    0    5 1077
##
## Overall Statistics
##
##                Accuracy : 0.9946
##                  95% CI : (0.9923, 0.9963)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9931
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9947   0.9864   0.9927   0.9991
## Specificity            0.9998   0.9992   0.9986   0.9970   0.9990
## Pos Pred Value         0.9994   0.9965   0.9932   0.9844   0.9954
## Neg Pred Value         0.9991   0.9987   0.9971   0.9986   0.9998
## Prevalence             0.2850   0.1939   0.1755   0.1624   0.1832
## Detection Rate         0.2843   0.1929   0.1732   0.1613   0.1830
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9987   0.9969   0.9925   0.9948   0.9990
```

```
accuracy <- postResample(predictRf, as.factor(testData$classe))
accuracy
```

```
##  Accuracy     Kappa
## 0.9945624 0.9931216
```

```
oose <- 1 - as.numeric(confusionMatrix(as.factor(testData$classe), predictRf)$overall[1])
oose
```

```
## [1] 0.005437553
```

So, the estimated accuracy of the model is 99.42% and the estimated out-of-sample error is 0.53%.

Predicting for Test Data Set

Now, we apply the model to the original testing data set downloaded from the data source. We remove the problem_id column first.

```
result <- predict(modelRf, pml_testing_cleaned[, -length(names(pml_testing_cleaned))])
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
```

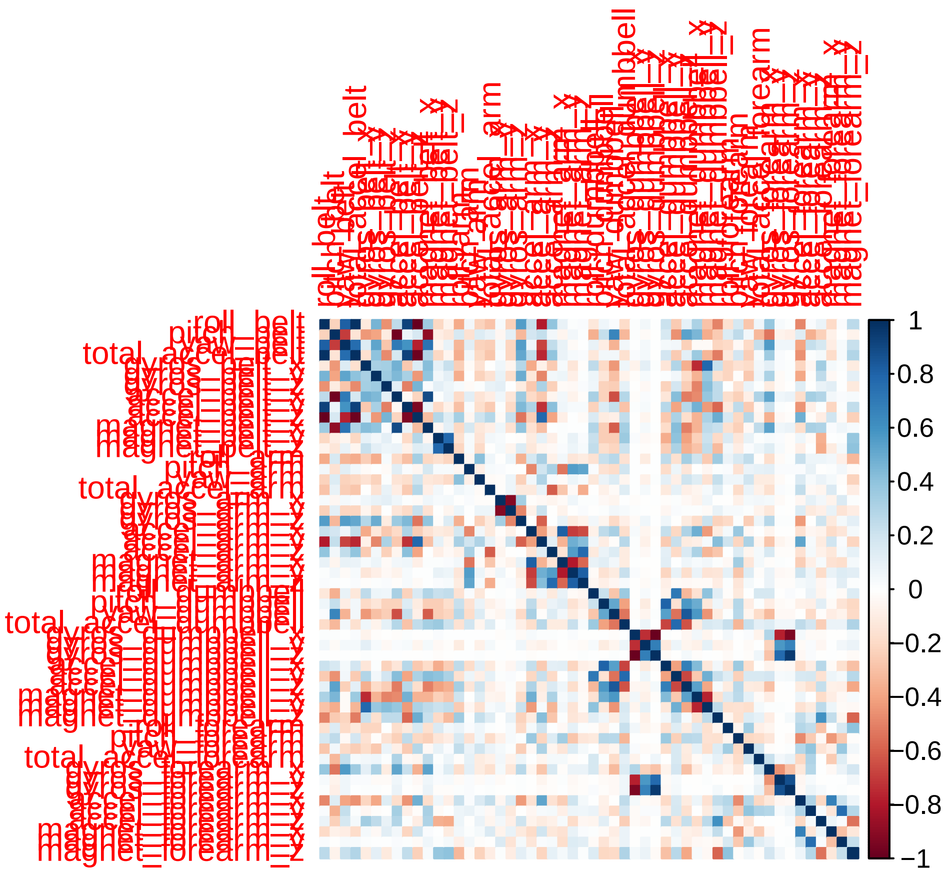
Conlusion

we find that the Accuracy of the Random Forest Model and error is better than the Decision Tree model. so we conclude that the random forest is the better model.

Appendix: Figures

1. Correlation Matrix Visualization

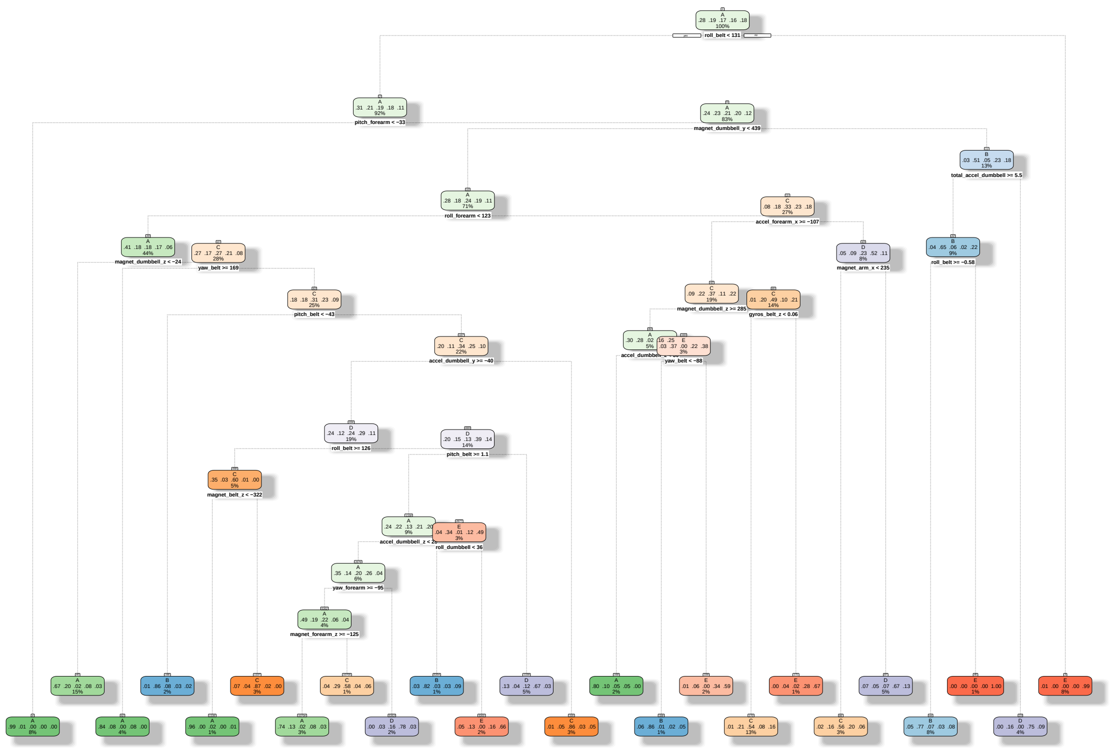
```
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, method="color")
```



2. Decision Tree Visualization

```r
treeModel <- rpart(classe ~ ., data=trainData, method="class")
fancyRpartPlot(treeModel)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2020−Oct−21 19:35:36 Andriy