ML-Project.R

Esmaeel Moradi

7/21/2019

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. 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.

Loadinmg packages, Reading data, and cleaning data

```
library(caret)
```

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

```
## Warning: package 'lattice' was built under R version 3.4.4
```

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

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

```
## Warning: package 'rattle' was built under R version 3.4.4
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.4.4
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## 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
```

```
setwd("/Users/esmaeel/Desktop/CorseraDataScienceLearning/ML/Project")
training = read.csv("pml-training.csv")
testing = read.csv("pml-testing.csv")
```

Now, we remove all the unnecessary columns and columns with NA's.

```
# the first 7 rows do not matter, so we remove them
training_clean <- training[,8:length(colnames(training))]
testing_clean <- testing[,8:length(colnames(testing))]

# There are some columns with majority NA's, these columns will not play a roll in prediction
training_clean <- training_clean[, colSums(is.na(training_clean)) == 0]
testing_clean <- testing_clean[, colSums(is.na(testing_clean)) == 0]

# the columns with no variance in them are basically not playing a roll in prediction
nzv <- nearZeroVar(training_clean,saveMetrics=TRUE)
variable_zero <- sum(nzv$nzv)

if ((variable_zero>0)) {
        training_clean <- training_clean[,nzv$nzv==FALSE]
}
```

Cross Validation

To make sure that the test data set is not being used in our model training, we do cross validation in this project. The cleaned training data set is divided into two train and test data sets.

```
in_training_clean <- createDataPartition(training_clean$classe, p=0.70, list=F)
training_clean_train <- training_clean[in_training_clean, ]
training_clean_valid <- training_clean[-in_training_clean, ]
```

Prediction Model

Train the model

The training data set is being used to train a random forest. Also, we use a 5-fold cross-validation technique in this model. It means that we tend to divide our training data set into 5 sub sets and select one of them as

test and train the model on rest and then repeat the process for 5 times and present the average.

```
control_method <- trainControl(method="cv", 5)
rf_model <- train(classe ~ ., data=training_clean_train, method="rf",
                  trControl=control_method, ntree=251)
rf_model
```

```
## 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: 10991, 10990, 10990, 10990, 10987
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9909008  0.9884885
##   27    0.9916293  0.9894113
##   52    0.9874068  0.9840693
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Now let's predict the results:

```
rf_predict <- predict(rf_model, training_clean_valid)
confusionMatrix(training_clean_valid$classe, rf_predict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669    2    2    0    1
##          B   10 1128    1    0    0
##          C    0    5 1019    2    0
##          D    0    0   11  951    2
##          E    0    0    3    5 1074
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.99, 0.9946)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9940   0.9938   0.9836   0.9927   0.9972
```

```
## Specificity             0.9988   0.9977   0.9986   0.9974   0.9983
## Pos Pred Value          0.9970   0.9903   0.9932   0.9865   0.9926
## Neg Pred Value          0.9976   0.9985   0.9965   0.9986   0.9994
## Prevalence              0.2853   0.1929   0.1760   0.1628   0.1830
## Detection Rate          0.2836   0.1917   0.1732   0.1616   0.1825
## Detection Prevalence    0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy       0.9964   0.9958   0.9911   0.9950   0.9978
```

as we can see the accuracy is: 0.9917.

Run model on Test data set

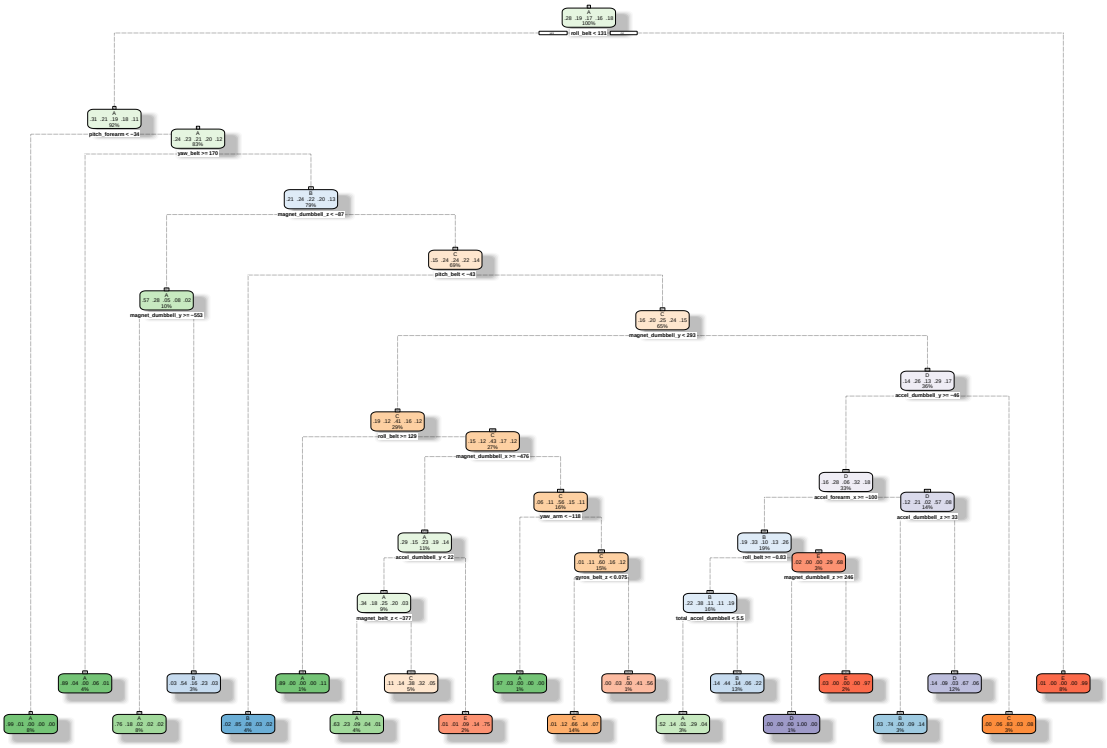
```
test_results <- predict(rf_model,
                  testing_clean[, -length(names(testing_clean))])
#test_results <- predict(rf_model, testing_clean)
#confusionMatrix(testing_clean$classe, test_results)
test_results
```

```
##  [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
```

Showing the decision tree:

```
tree_Model <- rpart(classe ~ ., data=training_clean_train, method="class")
fancyRpartPlot(tree_Model)
```

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



Rattle 2019−Jul−21 19:58:04 esmaeel