Final Project- Practical Machine Learning, Kevin Urbina

GitHub Repo:https://github.com/Scotturbina/Practical-Machine-learning-project/

Introduction:

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://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data source

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

Data Procesing

Loading Packages and importing data

```
#Loading required packages
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.3.3
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.3.3
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.3.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.3
```

```
## randomForest 4.6-12
```

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

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

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

```
library(repmis)
```

```
## Warning: package 'repmis' was built under R version 3.3.3
```

```
# Load data locally once you download it from URL's

#import the data from the URLs
trainurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
testurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
training <- source_data(trainurl, na.strings = c("NA", "#DIV/0!", ""), header
= TRUE)
```

```
## Downloading data from:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
```

```
## SHA-1 hash of the downloaded data file is:
## c7d3bd5499ad11292935ace3d69d92b2915be894
```

```
testing <- source_data(testurl, na.strings = c("NA", "#DIV/0!", ""), header =
TRUE)
```

```
## Downloading data from:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
```

```
## SHA-1 hash of the downloaded data file is:
## 7d3270a3e28102e16420acbce902a5f05286dee1
```

Training set has 160 variables as same as Testing set, what we are trying to predict is the outcome of classe in the training set.

Cleaning the data

Deleting predictor that contains missing values

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
```

Removing first seven predictors since they do not have much impact in the outcome of classe.

```
trainData <- training[, -c(1:7)]
testData <- testing[, -c(1:7)]
```

Now data is clean, both (Testing and training) has 57 variables

Data Split method

The data (trainData) is going to be splited in 70% training and 30% for validation, this in order to compute out of sample errors.

```
set.seed(661993)
inTrain <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
train <- trainData[inTrain, ]
valid <- trainData[-inTrain, ]
```

Prediction Algotihms Methods

We are going to use classification tress and random forest in order to predict the outcome

Classification trees

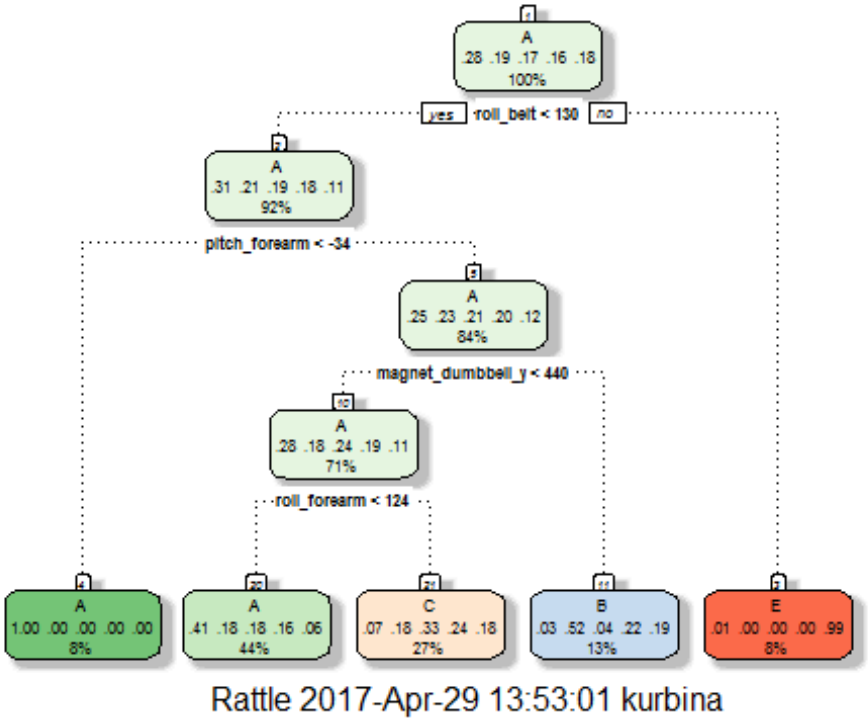
K-fold validation= 10

```
Ctrl_P <- trainControl(method = "cv", number = 10)

Fit<- train(classe ~ ., data = train, method = "rpart", trControl = Ctrl_P)

print(Fit, digits=4)
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12364, 12363, 12364, 12364, 12364, 12364, ...
```

```
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.03723  0.5085    0.3579
##   0.06069  0.4425    0.2527
##   0.11606  0.3155    0.0475
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03723.
```

```
fancyRpartPlot(Fit$finalModel)
```



Rattle 2017-Apr-29 13:53:01 kurbina

Looking at predictors results

```
Predict_CTS <- predict(Fit, valid)
#
Predictor_result <-confusionMatrix(valid$classe, Predict_CTS)
Predictor_result
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1513   23  135    0    3
##          B  480  365  294    0    0
##          C  469   30  527    0    0
##          D  456  186  322    0    0
##          E  152  151  300    0  479
```

```
##
## Overall Statistics
##
##                Accuracy : 0.4901
##                  95% CI : (0.4772, 0.5029)
##     No Information Rate : 0.5217
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3334
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4928  0.48344  0.33397       NA  0.99378
## Specificity            0.9428  0.84912  0.88414   0.8362  0.88840
## Pos Pred Value         0.9038  0.32046  0.51365       NA  0.44270
## Neg Pred Value         0.6303  0.91783  0.78370       NA  0.99938
## Prevalence             0.5217  0.12829  0.26814   0.0000  0.08190
## Detection Rate         0.2571  0.06202  0.08955   0.0000  0.08139
## Detection Prevalence   0.2845  0.19354  0.17434   0.1638  0.18386
## Balanced Accuracy      0.7178  0.66628  0.60905       NA  0.94109
```

Overall Statistics show us that model accuracy is 0.49 and Kappa 0.33 which tell us that this algorithm not work well to predict very well the outcome (We need 0.7 of Kappa statistics at least)

Random Forest

Lets try Random forest and evaluate the output

```
Fit_RandF <- train(classe ~ ., data = train, method = "rf", trControl =
Ctrl_P)

print(Fit_RandF, digits = 4)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12362, 12364, 12363, 12364, 12363, 12364, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9927    0.9908
##   27    0.9929    0.9910
##   52    0.9843    0.9801
```

```
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Predicting Outcomes

```
Predict_RandF <-predict(Fit_RandF, valid)

# Prediction result

result_RandF <- confusionMatrix(valid$classe, Predict_RandF)

result_RandF

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    0    2    0    0
##          B    9 1129    1    0    0
##          C    0    5 1017    4    0
##          D    0    0    5  958    1
##          E    2    1    4   10 1065
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.99, 0.9946)
##     No Information Rate : 0.286
##     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            0.9935   0.9947   0.9883   0.9856   0.9991
## Specificity            0.9995   0.9979   0.9981   0.9988   0.9965
## Pos Pred Value         0.9988   0.9912   0.9912   0.9938   0.9843
## Neg Pred Value         0.9974   0.9987   0.9975   0.9972   0.9998
## Prevalence             0.2860   0.1929   0.1749   0.1652   0.1811
## Detection Rate         0.2841   0.1918   0.1728   0.1628   0.1810
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9965   0.9963   0.9932   0.9922   0.9978
```

The accuracy of this model is very good as same as the Kappa statistic but it is computationally inefficient.

Predict on Testing set

The predicting is going to be done with random forest since was the best model.

```
TestPre<-predict(Fit_RandF, testData)
TestPre
```

```
##  [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
```