PML_Proj.

Yon Hai

9/29/2020

```r
options(knitr.duplicate.label = 'allow')
```

Summary

Below is a brief description of step-by-step process of prediction for Coursera's Practical Machine Learning project. That includes the process to fetch, clean and preprocess the data and build three types of prediction models to determine the best way to predict outcomes. Below of a brief discriotion of the background story on the data collection process taken from the project description.

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

Loading packages and data

```r
## packages
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(caret)
library(randomForest)
library(corrplot)
library(gbm)
library(dplyr)

pml_train <- read.csv('pml-training.csv', header=T)
pml_valid <- read.csv('pml-testing.csv', header=T)

dim(pml_train)
```

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

```r
dim(pml_valid)
```

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

This shows that we have 19622 observations to train our models and test the models on 20 cases on the valid data. First, We need to remove variables that have little to no significance in predicting the outcomes.

Clean Data: Remove incomplete and irrelevant variables

```r
pml_train  <- pml_train %>%
  select_if(~ !any(is.na(.)))%>% #incomplete variables
  select(-c(1:7))# irrelevant/identification variables

pml_valid <- pml_valid%>%
  select_if(~ !any(is.na(.))) %>%
  select(-c(1:7))

dim(pml_train)
```

```
## [1] 19622    86
```

```r
dim(pml_valid)
```

```
## [1] 20 53
```

Once we can remove the identification variables and incomplete variables, the training data has 86 variables and the test data has 53 variables.

Preparing the datasets for prediction

Preparing the data for prediction includes splitting the data into training and test data. Accordingly, 75% of the pml_train data to train the models and 25% of it to test the models. The models that performs well, with higher level of accuracy score, will be used to predict the outcomes in the pml_valid data.

```r
set.seed(2334)
Data_partition <- createDataPartition(pml_train$classe, p = 3/4)[[1]]
pml_train_Data <- pml_train[Data_partition , ]
pml_test_Data <- pml_train[-Data_partition , ]

dim(pml_train_Data)
```

```
## [1] 14718    86
```

```r
dim(pml_test_Data)
```

```
## [1] 4904   86
```

This produced 14718 observation in the training data and 4904 in the test data. We can further clean the data by removing variables with near near zero variance.

```r
NZV <- nearZeroVar(pml_train)
pml_train_Data <- pml_train_Data[, -NZV]
pml_test_Data  <- pml_test_Data[, -NZV]

dim(pml_train_Data)
```

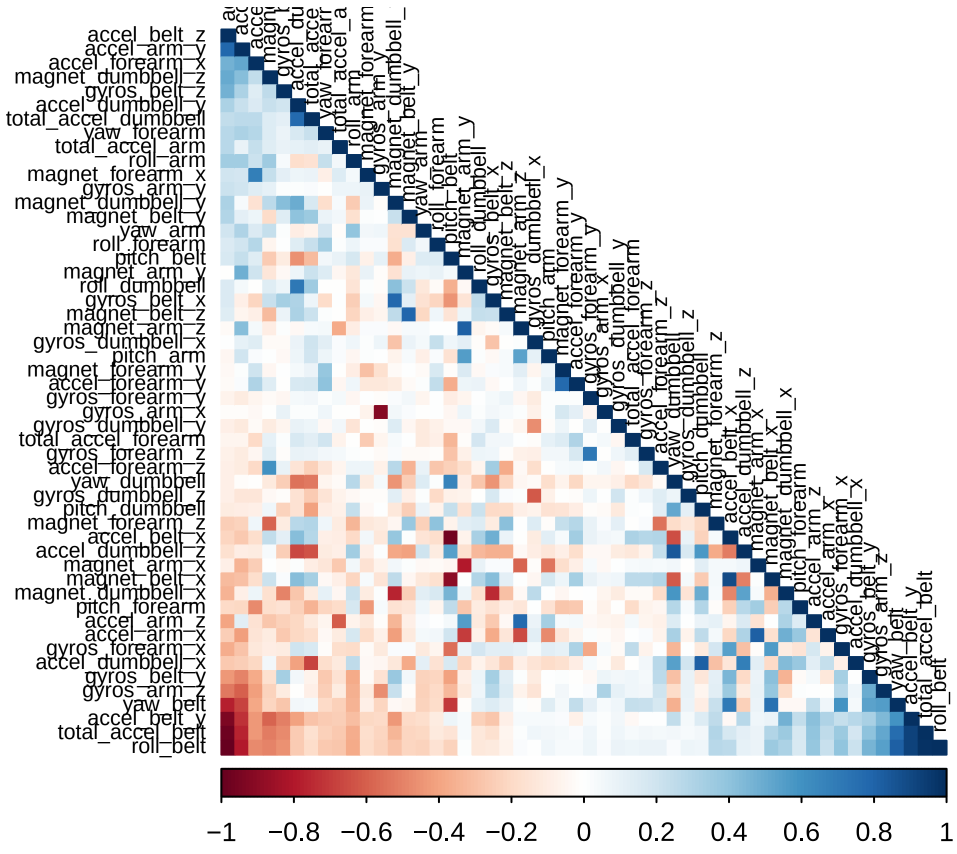
```
## [1] 14718    53
```

```r
dim(pml_test_Data)
```

```
## [1] 4904   53
```

Now we have 53 variables left in the dataset. We can now proceed to mapping the relationships between the variables in our dataset. ### correlation matrix

```r
cor_matrix <- cor(pml_train_Data[, -53])
corrplot(cor_matrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.7,  tl.col = rgb(0, 0, 0))
```



Squares with deep blue colors show strong positive association between the variables, where as squares with deep red colors shows negative association. We can also list the variables with high level of correlation as follows. Here are the variables with the positive correlation .75 and above.

```r
highlyCorrelated = findCorrelation(cor_matrix, cutoff=0.75)
names(pml_train_Data)[highlyCorrelated]
```

```
##  [1] "accel_belt_z"      "roll_belt"         "accel_arm_y"
##  [4] "accel_belt_y"      "yaw_belt"          "accel_dumbbell_z"
##  [7] "accel_belt_x"      "pitch_belt"        "magnet_dumbbell_x"
## [10] "accel_dumbbell_y"  "magnet_dumbbell_y" "accel_dumbbell_x"
## [13] "accel_arm_x"       "accel_arm_z"       "magnet_arm_y"
## [16] "magnet_belt_y"     "accel_forearm_y"   "gyros_arm_x"
```

Model Building

Prediction with Random Forest

Now, we will use three types of models to predicts the 20 cases of classe variables in our dataset. the models implemented below are Random Forests, Decision Tree and Generalized Boosted Mode. In each model, we will train the dataset using the plm_train_data and test them on pml_test_data. Confusion matrix will be used to visualize each prediction's accuracy level. This will help us determine the best performing model to apply to out pml_vaid data.

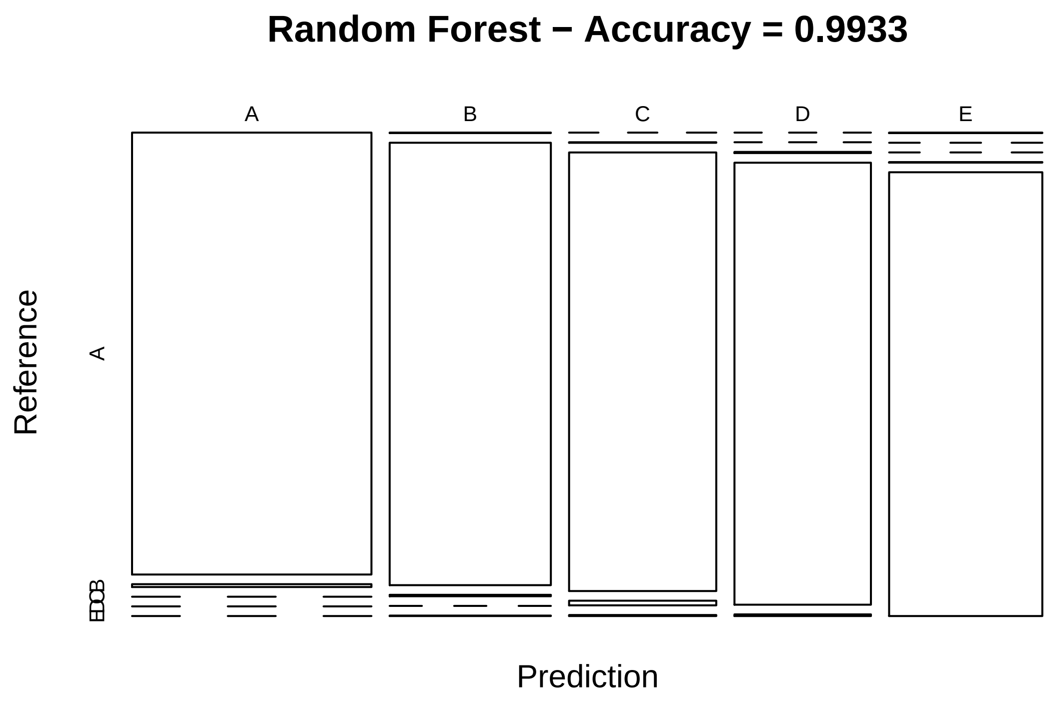
```r
set.seed(63456)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
mod_rf <- train(classe ~ ., data =pml_train_Data , method = "rf", trControl=controlRF)
```

```r
## prediction on test dataset

predict_mod_rf <- predict(mod_rf, pml_test_Data)
cm_rf <- confusionMatrix(predict_mod_rf, pml_test_Data$classe)
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    9    0    0    0
##          B    1  939    3    0    1
##          C    0    1  850    9    2
##          D    0    0    2  794    3
##          E    1    0    0    1  895
##
## Overall Statistics
##
##                Accuracy : 0.9933
##                  95% CI : (0.9906, 0.9954)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9915
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9895   0.9942   0.9876   0.9933
## Specificity            0.9974   0.9987   0.9970   0.9988   0.9995
## Pos Pred Value         0.9936   0.9947   0.9861   0.9937   0.9978
## Neg Pred Value         0.9994   0.9975   0.9988   0.9976   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1915   0.1733   0.1619   0.1825
## Detection Prevalence   0.2859   0.1925   0.1758   0.1629   0.1829
## Balanced Accuracy      0.9980   0.9941   0.9956   0.9932   0.9964
```

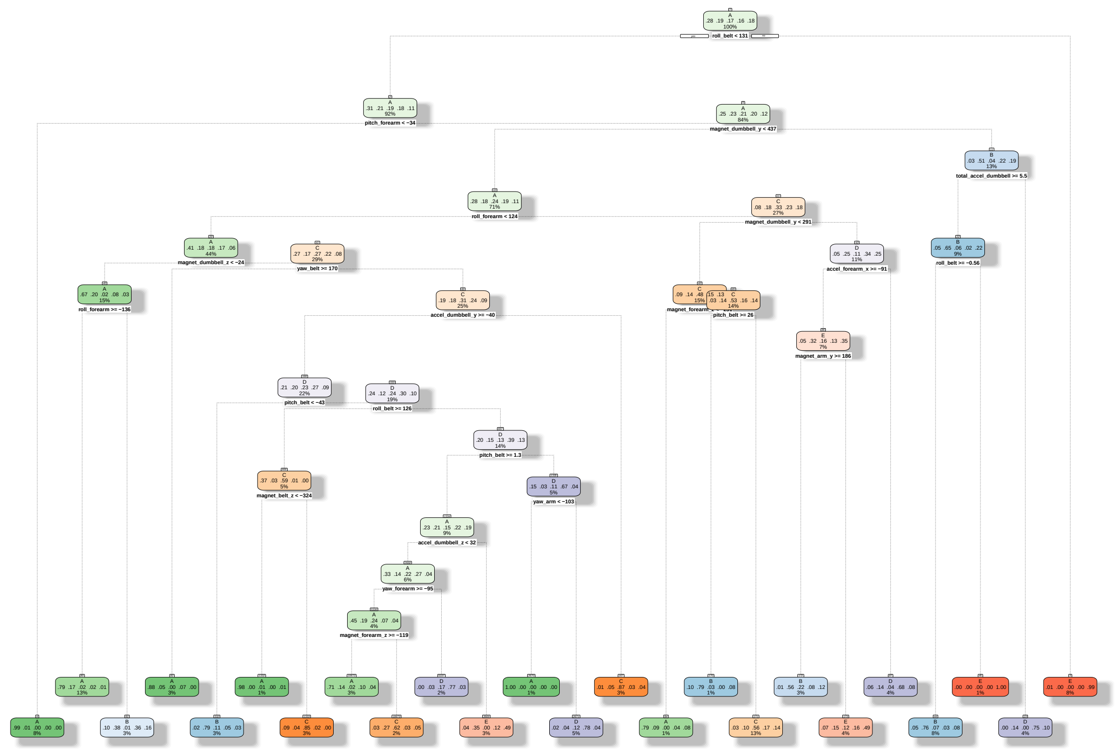
```r
plot(cm_rf$table, col = cm_rf$byClass,
main = paste("Random Forest - Accuracy =", round(cm_rf$overall['Accuracy'], 4)))
```



Prediction with Decision Trees

```r
set.seed(12345)
decision_Tree <- rpart(classe ~ ., data=pml_train_Data, method="class")
fancyRpartPlot(decision_Tree)
```

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



Rattle 2020−Oct−08 10:50:48 yhailu

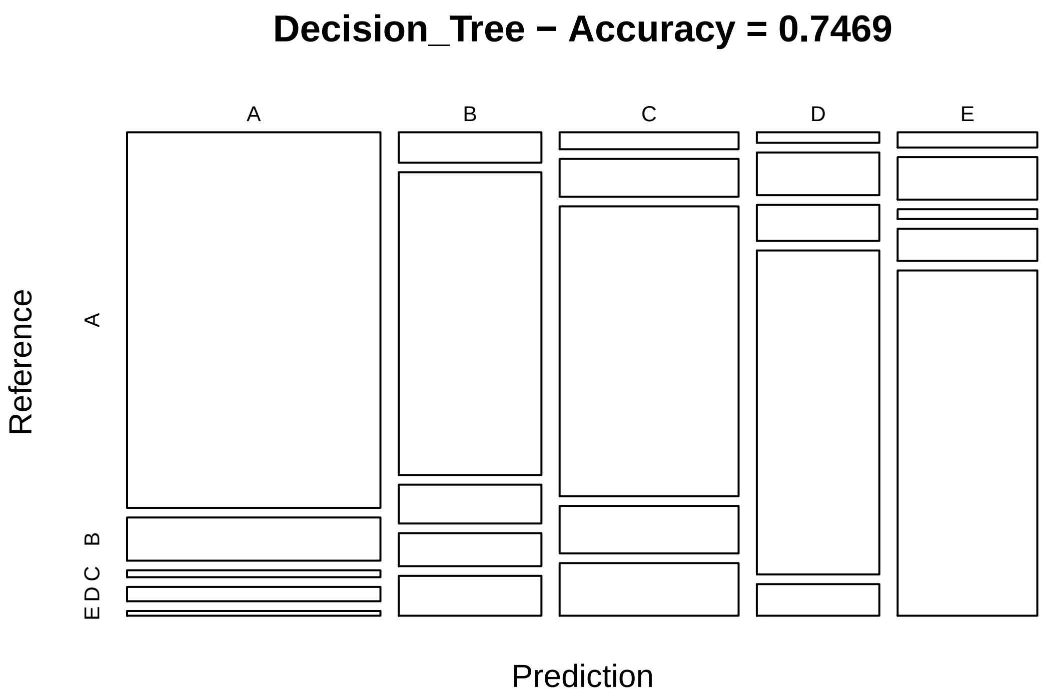
```
# prediction on test dataset

predict_decision_Tree <- predict(decision_Tree, pml_test_Data, type = "class")
cm_decision_Tree <- confusionMatrix(predict_decision_Tree, pml_test_Data$classe)
cm_decision_Tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1253  144   23   48   16
##          B   57  569   73   62   75
##          C   40   89  683  112  124
##          D   17   69   58  523   51
##          E   28   78   18   59  635
##
## Overall Statistics
##
##                Accuracy : 0.7469
##                  95% CI : (0.7345, 0.7591)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6793
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8982   0.5996   0.7988   0.6505   0.7048
## Specificity            0.9342   0.9325   0.9099   0.9524   0.9543
## Pos Pred Value         0.8443   0.6806   0.6517   0.7284   0.7763
## Neg Pred Value         0.9585   0.9066   0.9554   0.9329   0.9349
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2555   0.1160   0.1393   0.1066   0.1295
## Detection Prevalence   0.3026   0.1705   0.2137   0.1464   0.1668
## Balanced Accuracy      0.9162   0.7660   0.8543   0.8015   0.8295
```

```
plot(cm_decision_Tree$table, col = cm_decision_Tree$byClass,
main = paste("Decision_Tree - Accuracy =", round(cm_decision_Tree$overall['Accuracy'], 4)))
```



Prediction with Generalized Boosted Regression Models

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

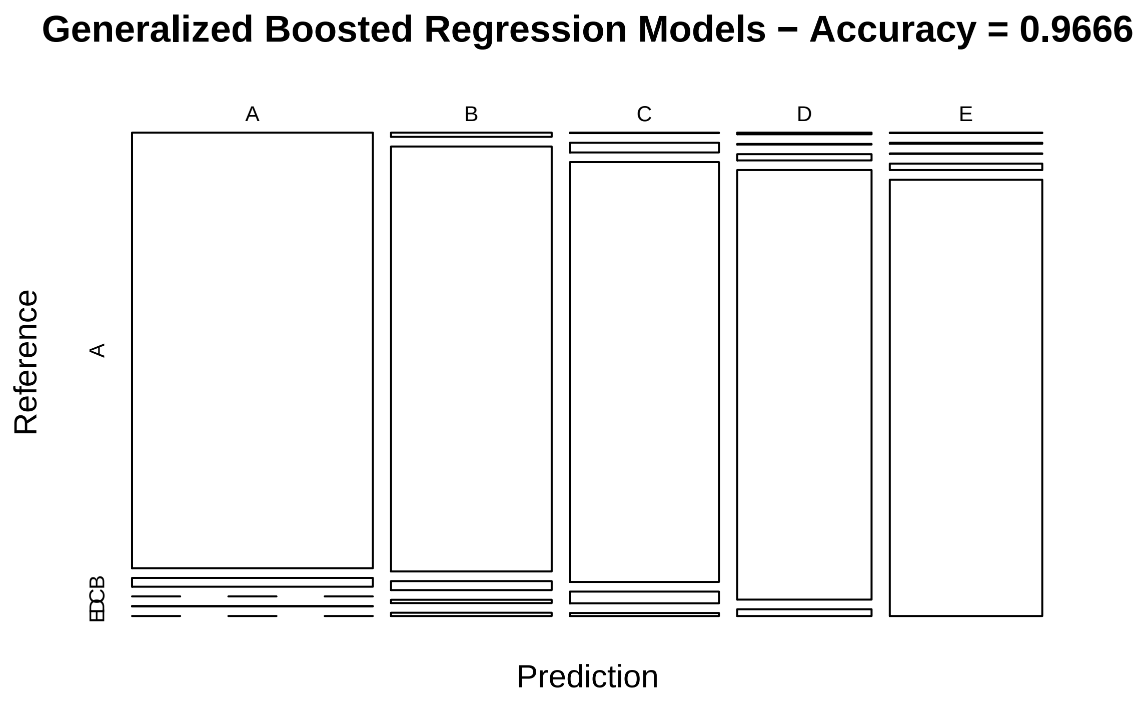
```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

```
# prediction on test dataset

predict_mod_GBM <- predict(mod_GBM, pml_test_Data)
cm_GBM <- confusionMatrix(predict_mod_GBM, pml_test_Data$classe)
cm_GBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1381   28    0    1    0
##          B    9  899   19    7    7
##          C    1   19  824   23    6
##          D    3    1   11  760   12
##          E    1    2    1   13  876
##
## Overall Statistics
##
##                Accuracy : 0.9666
##                  95% CI : (0.9611, 0.9714)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9577
##
##  Mcnemar's Test P-Value : 0.001886
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9900   0.9473   0.9637   0.9453   0.9723
## Specificity            0.9917   0.9894   0.9879   0.9934   0.9958
## Pos Pred Value         0.9794   0.9554   0.9439   0.9657   0.9810
## Neg Pred Value         0.9960   0.9874   0.9923   0.9893   0.9938
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2816   0.1833   0.1680   0.1550   0.1786
## Detection Prevalence   0.2875   0.1919   0.1780   0.1605   0.1821
## Balanced Accuracy      0.9908   0.9683   0.9758   0.9693   0.9840
```

```
plot(cm_GBM$table, col = cm_GBM$byClass,
main = paste("Generalized Boosted Regression Models - Accuracy =", round(cm_GBM$overall['Accuracy'], 4)
```



Model Selection

The above tests shows that accuracy level of Random Forest, Decision Trees, and Generalized Boosted Regression Models is 0.9933, 0.7469, and 0.9666. This shows that Random Forest model has the highest prediction ability with less than 0.001 out-of-sample error. on the other hand, Decision Tree model has the lowest accuracy level among the three models with around .25 out-of-sample errors. Accordingly, we will use Random Forest model to predict the 20 classe activity on the pml_valid_data (source dataset name: pml-testing)

```
predict_test_cases <- predict(mod_rf, newdata=pml_valid)
predict_test_cases
```

```
##  [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
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.