Practical Machine Learning (coursera) Assignment

Shahadat Hossain

30 Dec 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.Based on a dataset provide by HAR http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) we will try to train a predictive model.

We'll take the following steps:

- Explore and process the training data for the model(s)
- Model Selection and examination to find out the best performing model
- Predicting the test data based on the best fit model

Data Preparation

Downloading data

In the following, we will download datasets using the following links:

- Training data (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)
- Testing data (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The following codes will download data using the links:

```
# Downloading data
pml_training <- read.csv(file = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-trainin
g.csv")
pml_testing <- read.csv(file = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.
csv")

# Dimension of training data
tr_row <- dim(pml_training)[1]
tr_col <- dim(pml_training)[2]
```

The raw training data has 19622 rows and 160 columns.

Data cleaning

We will use testing dataset for validation purpose. Thus, in the following we will split the training data as training and testing data.

```
# Removing all the columns having atleast one missing value and all date/time/id related variabl
es
removeNA_df <- data.frame(na_count = colSums(is.na(pml_training) | pml_training == ""),
                          n_row = nrow(pml_training))
removeNA_df$var_names = rownames(removeNA_df)

removeNA_cols <- removeNA_df %>%
  filter(na_count/n_row > 0) %>%
  .$var_names

pml_trainingC <- pml_training %>%
  select(-c(removeNA_cols, grep("timestamp", names(.)), "X", "user_name")) %>%
  mutate(classe = as.factor(classe))

# Finding and removing highly inter-crrelated variables
classeIndex <- which(names(pml_trainingC) == "classe")

corMatrix <- cor(data.frame(data.matrix(pml_trainingC[, -classeIndex])))
highCor <- findCorrelation(corMatrix, cutoff = 0.9, exact = F)

pml_trainingC <- pml_trainingC[,-highCor]

# Removing columns with near zero variance
pml_trainingC <- pml_trainingC[, -nearZeroVar(pml_trainingC)]

trC_row <- dim(pml_trainingC)[1]
trC_col <- dim(pml_trainingC)[2]
```

As a part of cleaning, the above codes removes all the variables that have at least one missing values. Later, we have removed all the variables that have more that 90% correlation with other variable(s). Finally, we have dropped variables having almost zero variance. All following all the steps the final cleaned training dataset has 19622 rows and 47 columns.

Partitioning data sets

In the following, we have split the training raw datasets into another training data (have 70% of the observations) and testing data (having the rest 30% of the observations) using `caret` package.

```
# Data partition

inTrain <- createDataPartition(y = pml_trainingC$classe, p = 0.7, list = FALSE)
training_pml <-pml_trainingC[inTrain, ]
testing_pml <- pml_trainingC[-inTrain, ]
```

Model selection

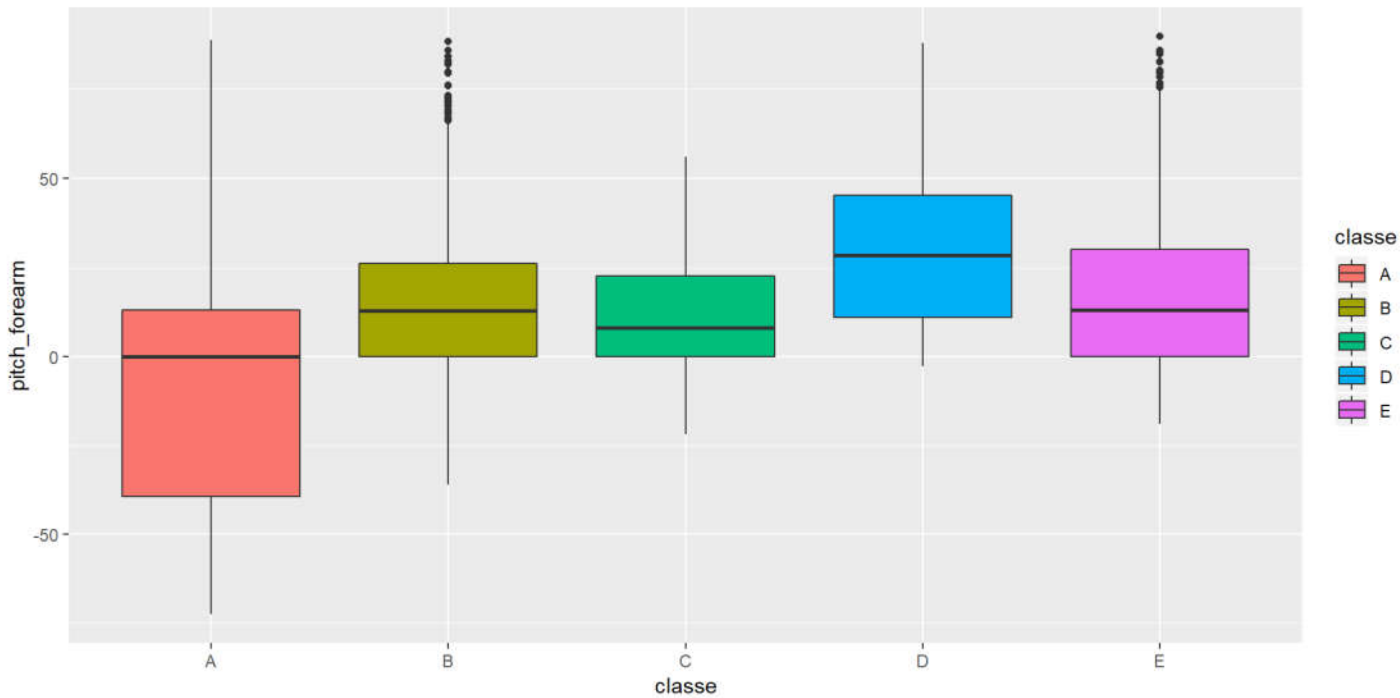
Initially, we have identified the correlation among the explanatory `classe` variable with other explanatory variables from `training_pml` dataset. The following codes find out the correlations:

```
bestCorr <- as.data.frame(as.table(cor(x = data.matrix(training_pml[,-classeIndex]),
                                       y = as.numeric(training_pml$classe)))) %>%
  filter(abs(Freq) > 0.3)
bestCorr
```

```
##            Var1 Var2      Freq
## 1 pitch_forearm    A 0.3454254
## 2        classe    A 1.0000000
```

We have found only `pitch_forearm` variable which has `corr > 0.3` with `classe` variable. In graph, we did not find the similar pattern.

```
plot1 <- ggplot(training_pml) +
  geom_boxplot(aes(x = classe, y = pitch_forearm, fill = classe))
plot1
```



Naive Bayes

```
# Naive Bayes
modFit_nb <- train(classe ~ .,
                   data = training_pml,
                   method = "nb",
                   trControl = trCtrl)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting fL = 0, usekernel = TRUE, adjust = 1 on full training set
```

```
pred_nb <- predict(modFit_nb, testing_pml)
accuracy_nb <- confusionMatrix(pred_nb, testing_pml$classe)$overall['Accuracy']
```

The accuracy of *Naive Bayes*`_ models is : 0.7610875.

Boosted Logistic Regression

```
# Boosted Logistic Regression
modFit_logbst <- train(classe ~ .,
                   data = training_pml,
                   method = "LogitBoost",
                   trControl = trCtrl)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting nIter = 31 on full training set
```

```
pred_logbst <- predict(modFit_logbst, testing_pml)
accuracy_logbst <- confusionMatrix(pred_logbst, testing_pml$classe)$overall['Accuracy']
accuracy_logbst
```

```
##  Accuracy
## 0.9435422
```

The accuracy of *Boosted Logistic Regression* models is : 0.9435422.

Stochastic Gradient Boosting

```
# Stochastic Gradient Boosting
modFit_gbm <- train(classe ~ .,
                       data = training_pml,
                       method = "gbm",
                       trControl = trCtrl)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting n.trees = 150, interaction.depth = 3, shrinkage = 0.1, n.minobsinnode = 10 on full tr
aining set
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2113
##      2        1.4771             nan     0.1000    0.1492
##      3        1.3858             nan     0.1000    0.1199
##      4        1.3115             nan     0.1000    0.1093
##      5        1.2450             nan     0.1000    0.0836
##      6        1.1937             nan     0.1000    0.0756
##      7        1.1465             nan     0.1000    0.0903
##      8        1.0932             nan     0.1000    0.0689
##      9        1.0526             nan     0.1000    0.0558
##     10        1.0193             nan     0.1000    0.0646
##     20        0.7347             nan     0.1000    0.0297
##     40        0.4683             nan     0.1000    0.0174
##     60        0.3369             nan     0.1000    0.0054
##     80        0.2560             nan     0.1000    0.0053
##    100        0.2009             nan     0.1000    0.0033
##    120        0.1612             nan     0.1000    0.0021
##    140        0.1293             nan     0.1000    0.0013
##    150        0.1164             nan     0.1000    0.0019
```

```
pred_gbm <- predict(modFit_gbm, testing_pml)
accuracy_gbm <- confusionMatrix(pred_gbm, testing_pml$classe)$overall['Accuracy']
accuracy_gbm
```

```
##  Accuracy
## 0.9874257
```

The accuracy of *Stochastic Gradient Boosting* models is : 0.9874257.

CART

```
# CART
modFit_rpart <- train(classe ~ .,
                    data = training_pml,
                    method = "rpart",
                    trControl = trCtrl)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting cp = 0.034 on full training set
```

```
pred_rpart <- predict(modFit_rpart, testing_pml)
accuracy_rpart <- confusionMatrix(pred_rpart, testing_pml$classe)$overall['Accuracy']
accuracy_rpart
```

```
##  Accuracy
## 0.5316907
```

The accuracy of *CART* models is : 0.5316907.

Random Forest

```
# Random Forest
modFit_rf <- train(classe ~ .,
                   data = training_pml,
                   method = "rf",
                   trControl = trCtrl)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 24 on full training set
```

```
pred_rf <- predict(modFit_rf, testing_pml)
accuracy_rf <- confusionMatrix(pred_rf, testing_pml$classe)$overall['Accuracy']
accuracy_rf
```

```
##  Accuracy
## 0.9976211
```

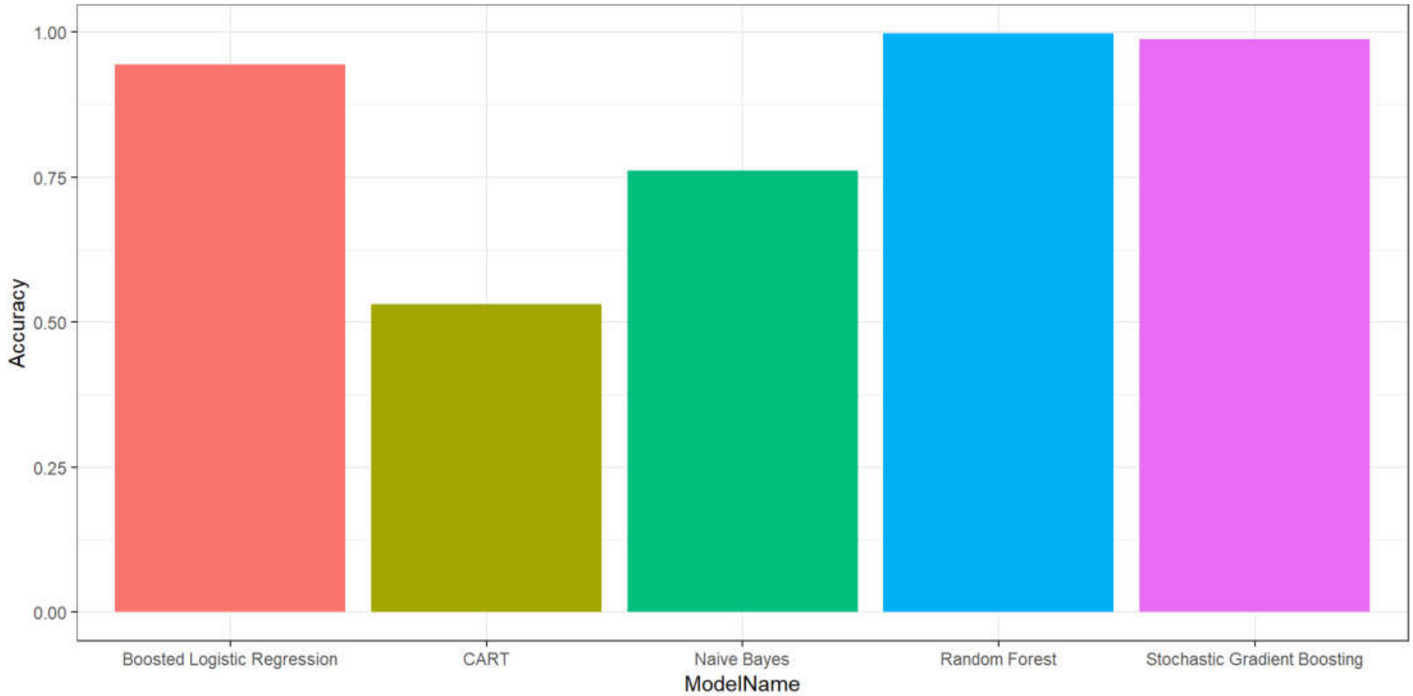
The accuracy of *Random Forest* models is : 0.9976211.

Model Performance

The following graph shows the accuracy of each of the model.

```
# Model Performance
modPerf <- data.frame(ModelName = c("Naive Bayes", "Boosted Logistic Regression",
                                       "Stochastic Gradient Boosting", "CART",
                                       "Random Forest"),
                      Accuracy = c(accuracy_nb, accuracy_logbst, accuracy_gbm,
                                   accuracy_rpart, accuracy_rf))

plot2 <- ggplot(modPerf, aes(x = ModelName, y = Accuracy)) +
  geom_bar(stat = "identity", aes(fill = ModelName)) +
  theme_bw() + theme(legend.position = "none")
plot2
```



From the graph, *Random Forest* is the best performing model, followed by Stochastic Gradient Boosting. Therefore, we will use Random Forest model for predicting from `plm_tesing` data.

Prediction

```
predVal <- predict(modFit_rf, pml_testing)

predVal_df <- data.frame(problem_id = paste0("Case: ",pml_testing$problem_id),
           Prediction = predVal)

predVal_df
```

```
##    problem_id Prediction
## 1     Case: 1          B
## 2     Case: 2          A
## 3     Case: 3          B
## 4     Case: 4          A
## 5     Case: 5          A
## 6     Case: 6          E
## 7     Case: 7          D
## 8     Case: 8          B
## 9     Case: 9          A
## 10   Case: 10          A
## 11   Case: 11          B
## 12   Case: 12          C
## 13   Case: 13          B
## 14   Case: 14          A
## 15   Case: 15          E
## 16   Case: 16          E
## 17   Case: 17          A
## 18   Case: 18          B
## 19   Case: 19          B
## 20   Case: 20          B
```