# Coursera Machine Learning Course Project

*Yue Li*

*Jan 21 2018*

***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://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

***Data Preparation***

First, we follow the instructions to download training and testing data from the provided url and then clean the data by removing invalid independent variables that have near zero values or mostly NA/blank:

```
library(lattice)
library(ggplot2)
library(caret)
library(e1071)

# set the URL for the download
UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# download the datasets
training <- read.csv(url(UrlTrain))
testing  <- read.csv(url(UrlTest))

# clean data & remove irrelevent predictors
NZV <- nearZeroVar(training)
training_clean <- training[, -NZV]
AllNA <- sapply(training_clean,function(x) mean(is.na(x) | x == "")) > 0.95
training_clean <- training_clean[, AllNA==FALSE]
training_clean <- training_clean[,-c(1:5)]
dim(training_clean)
```

```
## [1] 19622    54
```

Now we have 19622 records and 54 independent variables in the training data. To compare the model

performance created on the training data, we separate out a validation data subset from the training data:

```
# split the training data into a training set and a validation set
set.seed(1234)
inTrain <- createDataPartition(training_clean$classe, p = 0.6, list = FALSE)
trainset <- training_clean[inTrain,]
validset <- training_clean[-inTrain,]
dim(trainset)
```

```
## [1] 11776    54
```

```
dim(validset)
```

```
## [1] 7846   54
```

***Prediction Models***

Three methods are considered in building prediction models: random forest, generallized boosting model and support vector machine. The best model with highest accuracy rate in validation dataset would be used in final prediction quiz.

**1) Random Forest:**

```
# fit model using random forest
set.seed(2345)
controlrf <- trainControl(method="cv",number=5,verboseIter=FALSE)
mod_rf <- train(classe ~ ., data = trainset, method = "rf",trControl=controlrf)
mod_rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.35%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    0    0    0    1 0.0002986858
## B   10 2268    1    0    0 0.0048266784
## C    0    8 2045    1    0 0.0043816943
## D    0    0   11 1918    1 0.0062176166
## E    0    1    0    7 2157 0.0036951501
```

```
#predict on validation dataset
pred_rf <- predict(mod_rf, validset)

#Accuracy on validation dataset
confusionMatrix(pred_rf, validset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    4    0    0    0
##          B    0 1511    5    0    3
##          C    0    3 1362    6    0
##          D    0    0    1 1280    2
##          E    0    0    0    0 1437
##
## Overall Statistics
##
##                Accuracy : 0.9969
##                  95% CI : (0.9955, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9961
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9954   0.9956   0.9953   0.9965
## Specificity            0.9993   0.9987   0.9986   0.9995   1.0000
## Pos Pred Value         0.9982   0.9947   0.9934   0.9977   1.0000
## Neg Pred Value         1.0000   0.9989   0.9991   0.9991   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1926   0.1736   0.1631   0.1832
## Detection Prevalence   0.2850   0.1936   0.1747   0.1635   0.1832
## Balanced Accuracy      0.9996   0.9971   0.9971   0.9974   0.9983
```

```
confusionMatrix(pred_rf, validset$classe)$overall[1]
```

```
##  Accuracy
## 0.9969411
```

**2) Generallized Boosting Model:**

```
# fit model using gbm
set.seed(3456)
controlgbm <- trainControl(method="cv",number=5,verboseIter=FALSE)
mod_gbm <- train(classe ~ ., distribution="multinomial",data = trainset, method = "
gbm",trControl=controlgbm,verbose=FALSE)
mod_gbm$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 40 had non-zero influence.
```

```
#predict on validation dataset
pred_gbm <- predict(mod_gbm, validset)

#Accuracy on validation dataset
confusionMatrix(pred_gbm, validset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   25    0    0    0
##          B    1 1471   15    3    3
##          C    0   20 1341   24    4
##          D    0    2   11 1258   10
##          E    0    0    1    1 1425
##
## Overall Statistics
##
##                Accuracy : 0.9847
##                  95% CI : (0.9817, 0.9873)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9806
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9690   0.9803   0.9782   0.9882
## Specificity            0.9955   0.9965   0.9926   0.9965   0.9997
## Pos Pred Value         0.9889   0.9853   0.9654   0.9820   0.9986
## Neg Pred Value         0.9998   0.9926   0.9958   0.9957   0.9974
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1875   0.1709   0.1603   0.1816
## Detection Prevalence   0.2875   0.1903   0.1770   0.1633   0.1819
## Balanced Accuracy      0.9975   0.9828   0.9864   0.9874   0.9939
```

```
confusionMatrix(pred_gbm, validset$classe)$overall[1]
```

```
##  Accuracy
## 0.9847056
```

**3) Support Vector Machine:**

```
# fit model using svm
set.seed(4567)
mod_svm <- svm(classe ~ ., data = trainset)

#predict on validation dataset
pred_svm <- predict(mod_svm, validset)

#Accuracy on validation dataset
confusionMatrix(pred_svm, validset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2209  103    3    2    0
##          B    5 1374   41    0    8
##          C   18   38 1305  120   39
##          D    0    1   17 1163   28
##          E    0    2    2    1 1367
##
## Overall Statistics
##
##                Accuracy : 0.9454
##                  95% CI : (0.9402, 0.9504)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9309
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9897   0.9051   0.9539   0.9044   0.9480
## Specificity            0.9808   0.9915   0.9668   0.9930   0.9992
## Pos Pred Value         0.9534   0.9622   0.8586   0.9620   0.9964
## Neg Pred Value         0.9958   0.9776   0.9900   0.9815   0.9884
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2815   0.1751   0.1663   0.1482   0.1742
## Detection Prevalence   0.2953   0.1820   0.1937   0.1541   0.1749
## Balanced Accuracy      0.9852   0.9483   0.9604   0.9487   0.9736
```

```
confusionMatrix(pred_svm, validset$classe)$overall[1]
```

```
##  Accuracy
## 0.9454499
```

***Compare Accuracy***

```
accuracy_rf <- confusionMatrix(pred_rf, validset$classe)$overall[1]
accuracy_gbm <- confusionMatrix(pred_gbm, validset$classe)$overall[1]
accuracy_svm <- confusionMatrix(pred_svm, validset$classe)$overall[1]
accuracy_compare <- data.frame(Algorithm = c("Random Forest", "Generalized Boosting Model", "Support Vector Machine"), Accuracy = c(accuracy_rf, accuracy_gbm, accuracy_svm))
accuracy_compare
```

```
##                   Algorithm  Accuracy
## 1             Random Forest 0.9969411
## 2 Generalized Boosting Model 0.9847056
## 3     Support Vector Machine 0.9454499
```

We can see that Random Rorest has the highest accuracy rate of 99.67%, followed by Generalized Boosting Model of 98.47%, the last one Support Vetor Machine has the lowest the accuracy rate of 94.54%.

***Apply Prediction Model***

Apply the random forest model on testing data, the answers are:

```
predTesting <- predict(mod_rf, testing)
predTesting
```

```
##  [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
```