Practical Maching Learning - Course Project

Peace Liu

March 11, 2017

I. Overview

- The data used for this project are provided by the source groupware. Here I acknowledge them for being so generous to allow their data to be used for this project. This dataset contains personal activity features from all kinds of devices like *Jawbone Up, Nike FuelBand, and Fitbit*. In this project data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants will be used to predict the manner (**classe** variable in dataset) in which they did the exercise with all other variables as predictors.

- **classe** has 5 levels: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

- Three prediction methods (**Random Forests, Decision Tree and Generalized Boosted Model**) will be used to build the model in the training dataset. The best one with higher accuracy on the test dataset will be used for the prediction of the quiz.

II. Load and clean dataset

Load data

```
# set desired work directory
setwd("C:/Users/Fang Liu/Desktop/Fang Learning Video/Coursera-Data Science Specialzation By John Hopkins
# download training dataset
if (!file.exists("training.csv")) {
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", dest="training.
}
# download testing dataset
if (!file.exists("testing.csv")) {
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", dest="testing.csv
}
training <- read.csv("training.csv", header=TRUE, sep=",")  # 19622 obs of 160 variables
testing <- read.csv("testing.csv", header=TRUE, sep=",")    # 20 obs. of 160 variables
```

From training dataset, remove variables with Nearly Zero Variance, the variables with 80% NA and ID variables since these variables will not provide much power for prediction.

```
# remove nearly zero variables
library(caret); library(ggplot2)
```

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

```
NZV <- nearZeroVar(training)
training <- training[, -NZV]  # 19622 obs of 100 variables
# remove variables that are mostly NA
NAS <- sapply(training, function(x) mean(is.na(x))) > 0.8
training <- training[, NAS==FALSE]  # 19622 obs of 59 variables
# remove identification only variables (columns 1 to 5)
training <- training[, -(1:5)]      # 19622 obs of 54 variables
```

Now after cleaning up, the number of variables left for analysis has been reduced to 54.

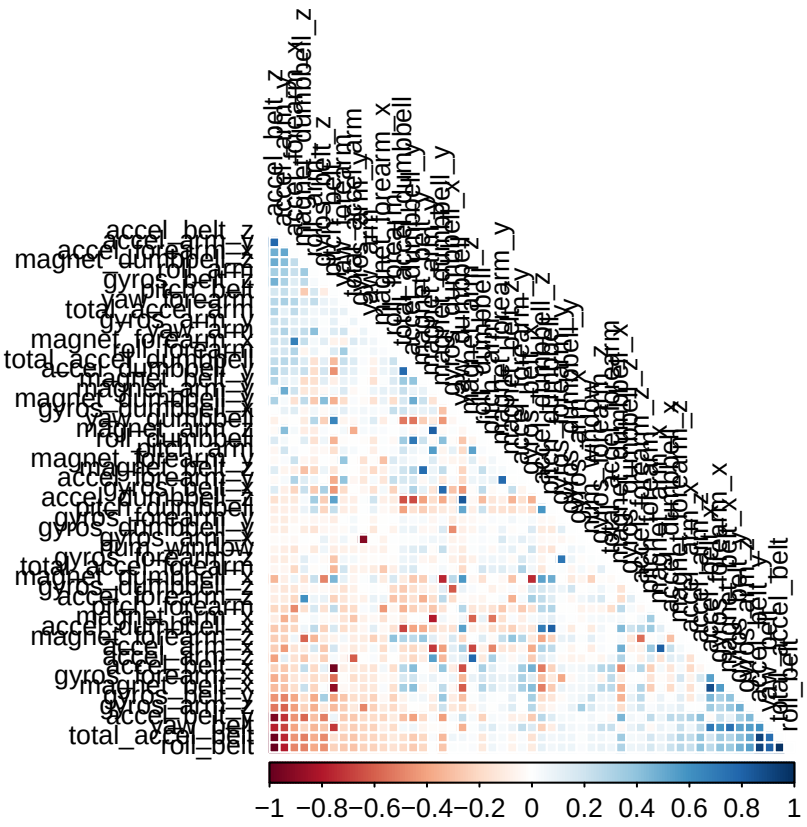
create a partition with the train and test dataset

```
inTrain <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]   # 13737 obs of 54 variables
TestSet  <- training[-inTrain, ]  # 5885 obs of 54 variables
```

III. Exploratory Analysis (Correlation Analysis)

Somtimes covariates can be highly correlated with each other. To do correlation analysis, leave out outcome variable in 54th column and only look at predictor variables and make self correlation=0 and plot corr plot

```
library(corrplot)
M <- cor(TrainSet[, -54]); diag(M) <- 0
corrplot(M, order = "FPC", method = "color", type = "lower", tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



From correlation matrix plot, there are few correlations among covariates.

IV. Prediction Model Building

Three methods will be applied to the trainset then predict on testset, pick the highest accuracy model to predict for quiz

1. Decision Trees

```
library(rpart)
set.seed(247)
treemodfit <- train(classe ~ ., method="rpart", data=TrainSet)
treemodfit$finalModel
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 129.5 12487 8627 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1110    9 A (0.99 0.0081 0 0 0) *
```

```
##      5) pitch_forearm>=-33.95 11377 8618 A (0.24 0.23 0.21 0.2 0.12)
##       10) num_window>=45.5 10859 8100 A (0.25 0.24 0.22 0.2 0.083)
##         20) num_window< 241.5 2438 1123 A (0.54 0.12 0.11 0.19 0.036) *
##         21) num_window>=241.5 8421 6067 B (0.17 0.28 0.25 0.2 0.097)
##           42) magnet_dumbbell_z< -26.5 2284 1215 A (0.47 0.35 0.048 0.12 0.0079) *
##           43) magnet_dumbbell_z>=-26.5 6137 4125 C (0.061 0.25 0.33 0.23 0.13)
##             86) magnet_dumbbell_x< -443.5 4376 2479 C (0.07 0.16 0.43 0.25 0.085) *
##             87) magnet_dumbbell_x>=-443.5 1761  932 B (0.039 0.47 0.065 0.18 0.24) *
##       11) num_window< 45.5 518   99 E (0 0 0 0.19 0.81) *
##    3) roll_belt>=129.5 1250   46 E (0.037 0 0 0 0.96) *
```

```
treemodfit
```

```
## CART
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa       Accuracy SD  Kappa SD
##   0.04343404  0.5093712  0.35709116  0.07876911   0.12539941
##   0.04506154  0.4929684  0.33142093  0.08512913   0.13643973
##   0.11779066  0.3139068  0.04459589  0.04086364   0.06070777
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.04343404.
```

```
# Predict on testset
preditTree <- predict(treemodfit, newdata = TestSet)
confusionM_tree <- confusionMatrix(preditTree, TestSet$classe)
confusionM_tree     # Accuracy=0.4882
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1470  485  147  332   57
##          B   36  345   36  110  208
##          C  140  309  843  475  168
##          D    0    0    0    0    0
##          E   28    0    0   47  649
##
## Overall Statistics
##
##                Accuracy : 0.5619
##                  95% CI : (0.5491, 0.5747)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.4351
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8781  0.30290   0.8216   0.0000   0.5998
## Specificity            0.7575  0.91783   0.7753   1.0000   0.9844
## Pos Pred Value         0.5901  0.46939   0.4357      NaN   0.8964
## Neg Pred Value         0.9399  0.84583   0.9537   0.8362   0.9161
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2498  0.05862   0.1432   0.0000   0.1103
## Detection Prevalence   0.4233  0.12489   0.3288   0.0000   0.1230
## Balanced Accuracy      0.8178  0.61036   0.7984   0.5000   0.7921
```

Decision tree generates a model with accuracy=0.4882

2. Random Forest

```
library(randomForest)
```

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

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

```
set.seed(247)
RFmodfit <- train(classe ~ ., method="rf", data=TrainSet, trControl=trainControl(method="cv", number=3,
RFmodfit$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: 0.22%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    1    0    0    0 0.0002560164
## B    6 2648    3    1    0 0.0037622272
## C    0    7 2388    1    0 0.0033388982
## D    0    0    6 2245    1 0.0031083481
## E    0    0    0    4 2521 0.0015841584
```

```
RFmodfit
```

```
## Random Forest
##
```

```
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9159, 9158, 9157
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD   Kappa SD
##    2    0.9923564  0.9903305  0.0013285703  0.001680565
##   27    0.9949772  0.9936460  0.0013096388  0.001657651
##   53    0.9934484  0.9917121  0.0008727435  0.001104892
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
# Predict on testset
preditRF <- predict(RFmodfit, newdata = TestSet)
confusionM_RF <- confusionMatrix(preditRF, TestSet$classe)
confusionM_RF     # Accuracy=0.998
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    0 1138    2    0    0
##          C    0    0 1024    4    0
##          D    0    0    0  960    2
##          E    1    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9969, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9991   0.9981   0.9959   0.9982
## Specificity            0.9998   0.9996   0.9992   0.9996   0.9998
## Pos Pred Value         0.9994   0.9982   0.9961   0.9979   0.9991
## Neg Pred Value         0.9998   0.9998   0.9996   0.9992   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1934   0.1740   0.1631   0.1835
## Detection Prevalence   0.2845   0.1937   0.1747   0.1635   0.1837
## Balanced Accuracy      0.9996   0.9994   0.9986   0.9977   0.9990
```

Random Forest generates a model with accuracy=0.998

3. Generalized Boosted Model

```
library(gbm); library(plyr)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

```
set.seed(247)
GBmodfit <- train(classe ~ ., method="gbm", data=TrainSet, verbose=FALSE, trControl=trainControl(method=
GBmodfit$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 42 had non-zero influence.
```

```
GBmodfit
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9159, 9158, 9157
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa      Accuracy SD
##   1                   50      0.7608638  0.6967102  0.009212271
##   1                  100      0.8314765  0.7866948  0.008111719
##   1                  150      0.8699861  0.8354689  0.002279904
##   2                   50      0.8823620  0.8510381  0.004090240
##   2                  100      0.9383415  0.9219675  0.002301283
##   2                  150      0.9622915  0.9522892  0.002485963
```

```
##   3                   50      0.9320088  0.9139298  0.003578979
##   3                  100      0.9716829  0.9641667  0.004608142
##   3                  150      0.9852227  0.9813043  0.003026112
##   Kappa SD
##   0.011524697
##   0.010232112
##   0.002859842
##   0.005225812
##   0.002913269
##   0.003145411
##   0.004527892
##   0.005844777
##   0.003830908
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
# Predict on testset
preditGB <- predict(GBmodfit, newdata = TestSet)
confusionM_GB <- confusionMatrix(preditGB, TestSet$classe)
confusionM_GB     # Accuracy=0.9876
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669    6    0    0    0
##          B    4 1125    7    7    8
##          C    0    7 1015   10    0
##          D    0    1    3  946    7
##          E    1    0    1    1 1067
##
## Overall Statistics
##
##                Accuracy : 0.9893
##                  95% CI : (0.9863, 0.9918)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9865
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9877   0.9893   0.9813   0.9861
## Specificity            0.9986   0.9945   0.9965   0.9978   0.9994
## Pos Pred Value         0.9964   0.9774   0.9835   0.9885   0.9972
## Neg Pred Value         0.9988   0.9970   0.9977   0.9963   0.9969
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
```

```
## Detection Rate         0.2836   0.1912   0.1725   0.1607   0.1813
## Detection Prevalence   0.2846   0.1956   0.1754   0.1626   0.1818
## Balanced Accuracy      0.9978   0.9911   0.9929   0.9895   0.9928
```

Generalized Boosted algorithm generates a model with accuracy=0.9876

V. Final Model selection and prediction to testing datase for quiz.

- Comparing 3 model predition accruacy, Random Forest has highest overall accuracy=0.998.
- The final random forests model contains 500 trees with 40 variables tried at each split. Estimated out of sample error rate for the random forests model is 0.04% from the final model.

- So predict the 20 quiz results (testing dataset) as shown below with randowm forest **RFmodfit**.

```
predict_test <- predict(RFmodfit, newdata = testing)
predict_test
```

```
##  [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
```