Prediction Assignment Writeup

Executive Summary

In this project, our goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to evaluate how well a person do a particular activity. Firstly, I will extract useful features from the training and testing dataset. Then I will use training dataset to train our models and envaluate the performance of each model accordingly. Finally, I will choose the best model to make predictions in our testing dataset.

Download and read both training and testing datasets

```
fileUrl1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
fileUrl2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
# download datasets
download.file(fileUrl2, destfile = "pml-testing.csv", method = "curl")
dataDownloaded <- date()
# read datasets
train <- read.csv("pml-training.csv",na.strings = c("NA", "#DIV/0!", ""))
validate <-read.csv("pml-testing.csv",na.strings = c("NA", "#DIV/0!", ""))
```

Data cleaning and processing

1. Check the dimensions of train and validate datasets

```
dim(train)
```

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

```
dim(validate)
```

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

2. Remove all columns that contains NA

Here we use "colSums(is.na()) == 0" to check whether there are NA in each column.

```
train <- train[,colSums(is.na(train)) == 0]
validate <- validate[,colSums(is.na(validate)) == 0]
```

Then we check the dimensions of train and validates datasets again.

```
dim(train)
```

```
## [1] 19622    60
```

```
dim(validate)
```

```
## [1] 20 60
```

3. Check the feature names and remove unrelated features

```
names(train)
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
##  [7] "num_window"           "roll_belt"            "pitch_belt"
## [10] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
## [13] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [16] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [19] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [22] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [25] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [28] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [31] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [34] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [37] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [40] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [43] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [46] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [49] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [52] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [55] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [58] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

Since our datasets has no time-dependence, so we decided to remove the first 7 features.

```
train <- train[,-(1:7)]
validate <- validate[,-(1:7)]
```

Data Partitioning

Here i decided to seperate train dataset into two parts, 60% of them will be used to train our models, while 40% of them will be used to evaluate our modles.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

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

```
inTrain <- createDataPartition(y = train$classe, p = 0.6, list = FALSE)
training <- train[inTrain,]
testing <- train[-inTrain,]
```

Use three machine learning strategies to train models

1. Classification Trees

```
library(rpart)
set.seed(123)
```

```
#modFit_rpart <- train(classe ~ ., method = "rpart", data = training,
#                      trControl=trainControl(method="cv",number = 10),
#                      tuneGrid=data.frame(cp=0.01))
modFit_rpart <- rpart(classe ~ ., data = training, method = "class")
prediction_rpart <- predict(modFit_rpart, testing, type = "class")
confusionMatrix(prediction_rpart, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2041  344   46  134   82
##          B   94  828  168   62  265
##          C   50  198 1059  186  136
##          D   30  117   95  807   93
##          E   17   31    0   97  866
##
## Overall Statistics
##
##                Accuracy : 0.7139
##                  95% CI : (0.7037, 0.7239)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6357
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9144   0.5455   0.7741   0.6275   0.6006
## Specificity            0.8921   0.9069   0.9120   0.9489   0.9774
## Pos Pred Value         0.7711   0.5843   0.6501   0.7067   0.8566
## Neg Pred Value         0.9633   0.8927   0.9503   0.9286   0.9157
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2601   0.1055   0.1350   0.1029   0.1104
## Detection Prevalence   0.3374   0.1806   0.2076   0.1456   0.1289
## Balanced Accuracy      0.9032   0.7262   0.8431   0.7882   0.7890
```

We can see that the accuracy of this Classification Tree is 0.74.

2. Randon Forest

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
```

```
##
##     margin
```

```
set.seed(123)
#modFit_rf <- train(classe ~ ., method = "rf", data = training, prox = TRUE,
#                   trControl=trainControl(method="cv",number = 10))
modFit_rf <- randomForest(classe ~ ., data = training, method = "rf",
          importance = T, trControl = trainControl(method = "cv", number = 10))
prediction_rf <- predict(modFit_rf, testing)
confusionMatrix(prediction_rf, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228    9    0    0    0
##          B    0 1505   12    0    0
##          C    0    4 1354   10    1
##          D    0    0    2 1276    2
##          E    4    0    0    0 1439
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9925, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9914   0.9898   0.9922   0.9979
## Specificity            0.9984   0.9981   0.9977   0.9994   0.9994
## Pos Pred Value         0.9960   0.9921   0.9890   0.9969   0.9972
## Neg Pred Value         0.9993   0.9979   0.9978   0.9985   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1918   0.1726   0.1626   0.1834
## Detection Prevalence   0.2851   0.1933   0.1745   0.1631   0.1839
## Balanced Accuracy      0.9983   0.9948   0.9937   0.9958   0.9986
```

We can see taht the accuracy of the Random Forest is 0.99.

3. Boosting:

```
library(gbm)
```

```
## 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 2.1.3
```

```
set.seed(123)
modFit_boosting <- train(classe ~ ., method = "gbm", data = training, verbose = FALSE, trControl=trainC
prediction_boosting <- predict(modFit_boosting, testing)
confusionMatrix(prediction_boosting, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2196   47    0    3    3
##          B   24 1434   55    2   14
##          C    6   36 1286   36   13
##          D    4    0   24 1233   17
##          E    2    1    3   12 1395
##
## Overall Statistics
##
##                Accuracy : 0.9615
##                  95% CI : (0.957, 0.9657)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9513
##  Mcnemar's Test P-Value : 1.361e-05
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9447   0.9401   0.9588   0.9674
## Specificity            0.9906   0.9850   0.9860   0.9931   0.9972
## Pos Pred Value         0.9764   0.9379   0.9339   0.9648   0.9873
## Neg Pred Value         0.9936   0.9867   0.9873   0.9919   0.9927
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2799   0.1828   0.1639   0.1572   0.1778
## Detection Prevalence   0.2866   0.1949   0.1755   0.1629   0.1801
## Balanced Accuracy      0.9872   0.9648   0.9630   0.9760   0.9823
```

We can see that the accuracy of this boosting is 0.96.

Make predictions

Here we choose Random Forest as our model because it has the highest accuracy. We can see the prediction of "classe" in validate dataset in the end.

```
prediction_validate <- predict(modFit_rf, newdata=validate)
prediction_validate
```

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