Prediction Analysis Process Writeup

1. Load package and Set seed

```
library(caret)

set.seed(12345)
```

2. Read in Data

NOTE: treat blank and invalid data (e.g., #DIV/0!) as **na**

```
url_train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

url_test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

data_train <- read.csv(url_train, na.strings = c("", "NA", "#DIV/0!"))

data_test <- read.csv(url_test, na.strings = c("", "NA", "#DIV/0!"))
```

3. Clean Data

1) delete columns that contains NA data

```
training<-data_train[,colSums(is.na(data_train)) == 0]

testing<-data_test[,colSums(is.na(data_test)) == 0]
```

2) delete columns (1 to 7) that are not related to the prediction

```
colnames(training)
```

```
[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"
```

```
training<-training[,-c(1:7)]
```

```
testing <-testing[,-c(1:7)]
```

4. Split Data into Train and Validation Sets

```
inTrain <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
```

```
InTraining <- training[inTrain, ]
```

```
InTesting <- training[-inTrain, ]
```

5. Train Different Models

1) Decision Tree

```
model_dt <- train(classe ~ ., data=InTraining, method="rpart")
```

```
predict_dt <- predict(model_dt, InTesting)
```

```
confusionMatrix(as.factor(InTesting$classe), predict_dt)
```

```
Confusion Matrix and Statistics

Reference
Prediction A B C D E
A 1252 30 90 0 23
B 396 317 236 0 0
C 434 24 397 0 0
D 343 151 310 0 0
E 114 132 229 0 426

Overall Statistics

Accuracy : 0.4878
95% CI : (0.4737, 0.5019)
No Information Rate : 0.5177
P-Value [Acc > NIR] : 1

Kappa : 0.3306

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: A Class: B Class: C Class: D Class: E
Sensitivity 0.4931 0.48471 0.31458 NA 0.94878
Specificity 0.9395 0.85129 0.87424 0.8361 0.89338
```

Pos Pred Value 0.8975 0.33404 0.46433 NA 0.47281
Neg Pred Value 0.6332 0.91479 0.78637 NA 0.99425
Prevalence 0.5177 0.13336 0.25734 0.0000 0.09156
Detection Rate 0.2553 0.06464 0.08095 0.0000 0.08687
Detection Prevalence 0.2845 0.19352 0.17435 0.1639 0.18373
Balanced Accuracy 0.7163 0.66800 0.59441 NA 0.92108

2) Random Forest

```
model_rf <- train(classe ~ ., data=InTraining, method="rf")

predict_rf <- predict(model_rf, InTesting)

confusionMatrix(as.factor(InTesting$classe), predict_rf)
```

Confusion Matrix and Statistics

Reference
Prediction A B C D E
A 1395 0 0 0 0
B 1 947 1 0 0
C 0 6 849 0 0
D 0 0 14 785 5
E 0 0 0 0 901

Overall Statistics

Accuracy : 0.9945
95% CI : (0.992, 0.9964)
No Information Rate : 0.2847
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.993

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: A Class: B Class: C Class: D Class: E
Sensitivity 0.9993 0.9937 0.9826 1.0000 0.9945
Specificity 1.0000 0.9995 0.9985 0.9954 1.0000
Pos Pred Value 1.0000 0.9979 0.9930 0.9764 1.0000
Neg Pred Value 0.9997 0.9985 0.9963 1.0000 0.9988
Prevalence 0.2847 0.1943 0.1762 0.1601 0.1847
Detection Rate 0.2845 0.1931 0.1731 0.1601 0.1837
Detection Prevalence 0.2845 0.1935 0.1743 0.1639 0.1837
Balanced Accuracy 0.9996 0.9966 0.9906 0.9977 0.9972

Validation results: though Random Forest took a much longer time to train, it results in a much higher accuracy of perdiction, so I will use this model to predict the testing data set.

6. Prediction on testing data

predict(model_rf, testing)

```
[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
```