# Markdown

Ji

2022-11-05

# Prediction Assignment Writeup

## Introduction

Goal of the project:
1. Use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise (“classe” variable in the training set).
2. Use model to predict the testing data.

## Data source

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

## Data cleaning

Load library, create file path and download data. Read the data, make blank data appear as NA.

```
library(caret)
```

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

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

```
library(ggplot2)

if(!file.exists("./raw_data")){
  dir.create("./raw_data")
}
url1<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url2<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(url1, "./raw_data/training.csv")
download.file(url2, "./raw_data/testing.csv")
training<-read.csv("./raw_data/training.csv",na.strings="")
testing<-read.csv("./raw_data/testing.csv",na.strings="")
str(training)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323
084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484
323 484434 ...
##  $ cvtd_timestamp          : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "0
5/12/2011 11:23" ...
##  $ new_window              : chr  "no" "no" "no" "no" ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4
...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : chr  NA NA NA NA ...
##  $ kurtosis_picth_belt     : chr  NA NA NA NA ...
##  $ kurtosis_yaw_belt       : chr  NA NA NA NA ...
##  $ skewness_roll_belt      : chr  NA NA NA NA ...
##  $ skewness_roll_belt.1    : chr  NA NA NA NA ...
##  $ skewness_yaw_belt       : chr  NA NA NA NA ...
##  $ max_roll_belt           : chr  "NA" "NA" "NA" "NA" ...
##  $ max_picth_belt          : chr  "NA" "NA" "NA" "NA" ...
##  $ max_yaw_belt            : chr  NA NA NA NA ...
##  $ min_roll_belt           : chr  "NA" "NA" "NA" "NA" ...
##  $ min_pitch_belt          : chr  "NA" "NA" "NA" "NA" ...
##  $ min_yaw_belt            : chr  NA NA NA NA ...
##  $ amplitude_roll_belt     : chr  "NA" "NA" "NA" "NA" ...
##  $ amplitude_pitch_belt    : chr  "NA" "NA" "NA" "NA" ...
##  $ amplitude_yaw_belt      : chr  NA NA NA NA ...
##  $ var_total_accel_belt    : chr  "NA" "NA" "NA" "NA" ...
##  $ avg_roll_belt           : chr  "NA" "NA" "NA" "NA" ...
##  $ stddev_roll_belt        : chr  "NA" "NA" "NA" "NA" ...
##  $ var_roll_belt           : chr  "NA" "NA" "NA" "NA" ...
##  $ avg_pitch_belt          : chr  "NA" "NA" "NA" "NA" ...
##  $ stddev_pitch_belt       : chr  "NA" "NA" "NA" "NA" ...
##  $ var_pitch_belt          : chr  "NA" "NA" "NA" "NA" ...
##  $ avg_yaw_belt            : chr  "NA" "NA" "NA" "NA" ...
##  $ stddev_yaw_belt         : chr  "NA" "NA" "NA" "NA" ...
##  $ var_yaw_belt            : chr  "NA" "NA" "NA" "NA" ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
```

```
##  $ yaw_arm                : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm        : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm          : chr  "NA" "NA" "NA" "NA" ...
##  $ avg_roll_arm           : chr  "NA" "NA" "NA" "NA" ...
##  $ stddev_roll_arm        : chr  "NA" "NA" "NA" "NA" ...
##  $ var_roll_arm           : chr  "NA" "NA" "NA" "NA" ...
##  $ avg_pitch_arm          : chr  "NA" "NA" "NA" "NA" ...
##  $ stddev_pitch_arm       : chr  "NA" "NA" "NA" "NA" ...
##  $ var_pitch_arm          : chr  "NA" "NA" "NA" "NA" ...
##  $ avg_yaw_arm            : chr  "NA" "NA" "NA" "NA" ...
##  $ stddev_yaw_arm         : chr  "NA" "NA" "NA" "NA" ...
##  $ var_yaw_arm            : chr  "NA" "NA" "NA" "NA" ...
##  $ gyros_arm_x            : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y            : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z            : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x            : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y            : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z            : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x           : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y           : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z           : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm      : chr  NA NA NA NA ...
##  $ kurtosis_picth_arm     : chr  NA NA NA NA ...
##  $ kurtosis_yaw_arm       : chr  NA NA NA NA ...
##  $ skewness_roll_arm      : chr  NA NA NA NA ...
##  $ skewness_pitch_arm     : chr  NA NA NA NA ...
##  $ skewness_yaw_arm       : chr  NA NA NA NA ...
##  $ max_roll_arm           : chr  "NA" "NA" "NA" "NA" ...
##  $ max_picth_arm          : chr  "NA" "NA" "NA" "NA" ...
##  $ max_yaw_arm            : chr  "NA" "NA" "NA" "NA" ...
##  $ min_roll_arm           : chr  "NA" "NA" "NA" "NA" ...
##  $ min_pitch_arm          : chr  "NA" "NA" "NA" "NA" ...
##  $ min_yaw_arm            : chr  "NA" "NA" "NA" "NA" ...
##  $ amplitude_roll_arm     : chr  "NA" "NA" "NA" "NA" ...
##  $ amplitude_pitch_arm    : chr  "NA" "NA" "NA" "NA" ...
##  $ amplitude_yaw_arm      : chr  "NA" "NA" "NA" "NA" ...
##  $ roll_dumbbell          : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell         : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell           : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell : chr  NA NA NA NA ...
##  $ kurtosis_picth_dumbbell: chr  NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell  : chr  NA NA NA NA ...
##  $ skewness_roll_dumbbell : chr  NA NA NA NA ...
##  $ skewness_pitch_dumbbell: chr  NA NA NA NA ...
##  $ skewness_yaw_dumbbell  : chr  NA NA NA NA ...
##  $ max_roll_dumbbell      : chr  "NA" "NA" "NA" "NA" ...
##  $ max_picth_dumbbell     : chr  "NA" "NA" "NA" "NA" ...
##  $ max_yaw_dumbbell       : chr  NA NA NA NA ...
##  $ min_roll_dumbbell      : chr  "NA" "NA" "NA" "NA" ...
##  $ min_pitch_dumbbell     : chr  "NA" "NA" "NA" "NA" ...
##  $ min_yaw_dumbbell       : chr  NA NA NA NA ...
```

```
##  $ amplitude_roll_dumbbell : chr  "NA" "NA" "NA" "NA" ...
##   [list output truncated]
```

Training data is 19622 observations of 160 variables. A lot of columns contains missing values. Some columns has the string “NA” when the data is missing.
To deal with missing values, the string “NA” is first replaced by NA. Then, columns contains NA are removed from the data. The same cleaning steps are also applied to the testing data.

```
#replace "NA" with NA
training[training=="NA"]<-NA
#analyze NA
na_count<-table(colSums(is.na(training)),colnames(training))
#100 columns has 19216 NA while 60 columns has 0 NA
sum(colSums(is.na(training))==19216)
```

```
## [1] 100
```

```
sum(colSums(is.na(training))==0)
```

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

```
#remove columns with 19216 NA
training_v1<-training[, colSums(is.na(training)) == 0]
#remove column 1-7
training_v1<-training_v1[,8:60]

#do the same to the testing data
testing[testing=="NA"]<-NA
testing_v1<-testing[, colSums(is.na(testing)) == 0]
testing_v1<-testing_v1[,8:60]
```

There are 100 columns in the training data, each has 19216 missing observations. 60 columns in the training data has 0 missing observations. The 100 columns are removed from the training data, which reduced the variables to 60.
In addition, column 1-7 are not useful information, so they are also removed from both the training and testing data. There are altogether 52 possible variables for the prediction of Classe.

# Desicion tree model

This model building is a classification problem. So tree models would be very helpful. There are multiple variables to choose from, so a cross validation should be used for model selection.
A K-fold cross validation with K = 10 is used in this model building. The average accuracy of the finla model is 0.506, which is poor.

```
set.seed(100)
# value of K equal to 10
train_control <- trainControl(method = "cv", number = 10)
tree_model<-train(classe ~ ., method="rpart", data=training_v1,
                  trControl = train_control)
print(tree_model) #average accuracy=0.506, poor
```

```
## CART
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17659, 17661, 17659, 17659, 17661, 17659, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03567868  0.5060147  0.35483589
##   0.05998671  0.4675852  0.29502212
##   0.11515454  0.3316225  0.07197029
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03567868.
```

# Random forest model

For better model selection, a random forest model is built. Random forest can do randomization of both variables and samples, so it is more powerful for model selection. There is no need to do cross validation because random sampling is already contained in the random forest idea.

```
#random forest model, no need to do CV
rf_model <- train(classe ~ ., data = training_v1, method = "rf") #this will take some time
print(rf_model)
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 19622, 19622, 19622, 19622, 19622, 19622, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9927719  0.9908563
##   27    0.9925507  0.9905764
##   52    0.9854863  0.9816388
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

The random forest model has an accuracy of 0.99 with mtry = 2. This is a very high accuracy. The model is reliable, much better than the result of decision tree model.

# Prediction on the testing data

```
#tree model
predict_tree<-predict(tree_model,newdata = testing_v1)
print("Prediction of tree model:")
```

```
## [1] "Prediction of tree model:"
```

```
predict_tree
```

```
##  [1] C A C A A C C A A A C C C A C A A A A C
## Levels: A B C D E
```

```
#rf model
pred_rf <- predict(rf_model, testing_v1)
print("Prediction of rf model:")
```

```
## [1] "Prediction of rf model:"
```

```
pred_rf
```

```
##  [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
```

# Conclusion

1. The accuracy of decision tree model with 10-fold cross validation is 0.506.
2. The accuracy of random forest model is 0.99.
3. The prediction of the testing data using random forest model is B A B A A E D B A A B C B A E E A B B B.