# Machine learning project for data science

## Introduction

This analysis aims to build a predictive model for the Weight Lifting Exercises Dataset.

## Load R packages and download data file

First, download and read the files. Missing values or error values are set to NA.

```
library(data.table)
library(ggplot2)
library(lattice)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

```
library(randomForest)
```

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

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rpart)
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
"training.csv")
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv","
testing.csv")
trainingfile<-fread("/Users/daxinggao/training.csv", na.strings=c("NA","#DIV/0!",""))
testingfile<-fread("/Users/daxinggao/testing.csv", na.strings=c("NA","#DIV/0!",""))
```

# Preprocess data

Create training and testing sets from training files. Only consider the columns that have fewer than 60% NA. And the first seven columns are also discarded.

```r
set.seed(1234)
inTrain<-createDataPartition(y=trainingfile$classe,p=0.75,list=F)#create training and testing sets.
training<-trainingfile[inTrain]
testing<-trainingfile[-inTrain]
NAcol<-sapply(training,function(x){sum(is.na(x))/nrow(training)})#calculate number of NA for each column
NAcolN<-match(names(NAcol[NAcol>=0.6]),colnames(training))#columns have more than 60% NA
preprocess<-preProcess(training[,-c(1:7,NAcolN,ncol(training)),with=F],method=c("center","scale","knnImpute"))#preprocess data, discard first seven columns/columns have more than 60% NA/last column, impute missing data with knnImpute
trainx<-predict(preprocess,training[,-c(1:7,NAcolN,ncol(training)),with=F])
trainnew<-cbind(trainx,training[,ncol(training),with=F])
```

# Use random forest model to fit and calculate prediction accuracy

```r
model<-randomForest(factor(classe) ~ .,trainnew)
confusionMatrix(predict(model,trainx),trainnew$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
## 
## Overall Statistics
## 
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

# Preprocess and predict the testing data and calculate prediction accuracy

```
testx<-predict(preprocess,testing[,-c(1:7,NAcolN,ncol(testing)),with=F])
testnew<-cbind(testx,testing[,ncol(testing),with=F])
confusionMatrix(predict(model,testx),testnew$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    3    0    0    0
##          B    0  944    9    0    0
##          C    0    2  845    7    0
##          D    0    0    1  797    0
##          E    0    0    0    0  901
## 
## Overall Statistics
## 
##                Accuracy : 0.9955
##                  95% CI : (0.9932, 0.9972)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9943
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9883   0.9913   1.0000
## Specificity            0.9991   0.9977   0.9978   0.9998   1.0000
## Pos Pred Value         0.9979   0.9906   0.9895   0.9987   1.0000
## Neg Pred Value         1.0000   0.9987   0.9975   0.9983   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1925   0.1723   0.1625   0.1837
## Detection Prevalence   0.2851   0.1943   0.1741   0.1627   0.1837
## Balanced Accuracy      0.9996   0.9962   0.9930   0.9955   1.0000
```

# Predict data from testing file

```
testfilex<-predict(preprocess,testingfile[,-c(1:7,NAcolN,ncol(testingfile)),with=F])
predict(model,testfilex)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 
##  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
```

# Conlusion

The random forest model successfully predicts the Weight Lifting Exercises Dataset.