Classification of Human Activity Recognition

Xiang Wang, Master Student in Applied Statistics

Department of Statistics, Penn State University

December 11, 2015

## Abstract

Wearable devices such as watch, band, ring and so on from big tech companies are starting to gain some popularity in the mainstream customers. They track and monitor the human activities such as jogging, walking, sleeping and so on. The raw data were collected from accelerometers on the belt, forearm, arm, and dumbell of six young health participants in five different **manners**. The objective of this project is to build a model to precisely predict or classify the manner each person did the exercise. Four classification methods are used: multinomial logit regression (MR), linear discriminant analysis (LDA), quadratic discriminant analysis (QDA) and random forest (RF). Results show that **random forest** method yields significantly higher prediction accuracy, sensitivity and specificity than the other three methods. Thus, I would prefer to this method to predict potential manners with new testing data.

### 1. Introduction

As an MAS student, I am specifically interested in apply data mining skills to meet the industry need for dealing with big data. This is also my potential job target when I will graduate in May 2016. One of the emerging markets for high-tech industry is wearable devices which include watch, band, ring and so on. Wearable devices are starting to gain some popularity in the mainstream customers as companies such as *Apple*, *Jawbone*, *Nike*, and *Fitbit* have debut their wearable products. Now it is relatively not expensive to collect a large amount of data about personal activity. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks.

### 2. Raw Data

The raw data were collected from accelerometers on the belt, forearm, arm, and dumbell of six young health participants. They were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different **manners** (the last variable "classe" in the data set): 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**).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. All participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

More information about the data is available from the website here: <http://groupware.les.inf.puc-rio.br/har> (see the section on the Weight Lifting Exercise Dataset). The original data for this project are available here: <http://groupware.les.inf.puc-rio.br/static/WLE/WearableComputing_weight_lifting_exercises_biceps_curl_variations.csv>. **Reference:** Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

### 3. Goal and Procedures

The goal of this project is to build a model to precisely predict or classify the manner each person did the exercise in the validation data. The specific procedures will be as follows:

* Raw data preparation. I will do necessary data cleaning and select ONLY the variables I may be interested in. Split the processed data into two parts (3:1 ratio): train data (70%) and validation data (30%).
* Preparation of new data for model selection. Principal component analysis was conducted to reduce the dimension of data before fitting the models since it has a large amount of predictor variables. Select appropriate amount of PCs (~85% variance explained). Standarize the training data and then standarize the validation data according to the mean and standard deviation of training data.
* Model building and comparison. The four classification methods to be used are: multinomial logit regression (MR), linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), and random forest (RF; I will explain this method in detail in 5.4). The training data will be used to fit different models from these methods; the validation data set will be used to assess how each model performs using the confusion matrix with five classes.

### 4. Data Preparation

### 4.1 Raw Data Preparation

data <- read.csv("WLE.csv")  
names(data); dim(data)

By running str(data) (not show the result due to space), we can see many numeric variables (for example, kurtosis\_yaw\_belt) are read as factor due to some characters ("#DIV/0!"); in addition, some variables have a lot of missing values. First, I need to reload the data by ignoring "#DIV/0!" values and change these variables to numeric values for the specified columns (i.e. column 8 to the second last column).

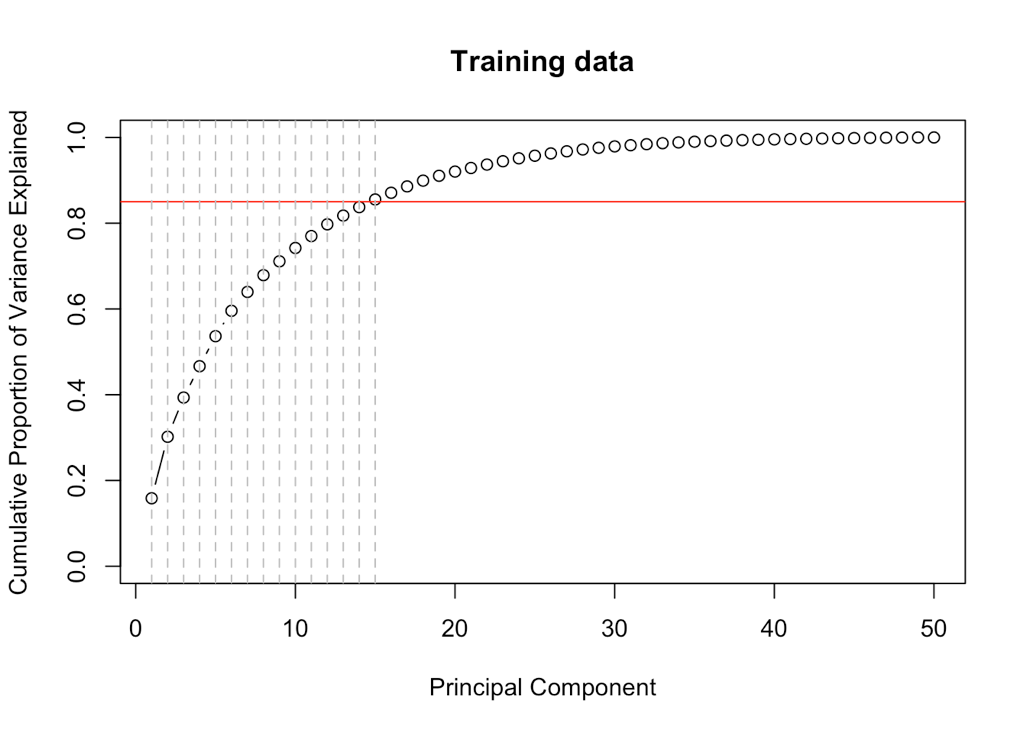
data <- read.csv("WLE.csv", header=TRUE, sep=',', na.strings=c("#DIV/0!"))  
data1 <- data   
for(i in c(7:ncol(data1)-1)) {  
 data1[,i]=as.numeric(as.character(data1[,i]))  
}

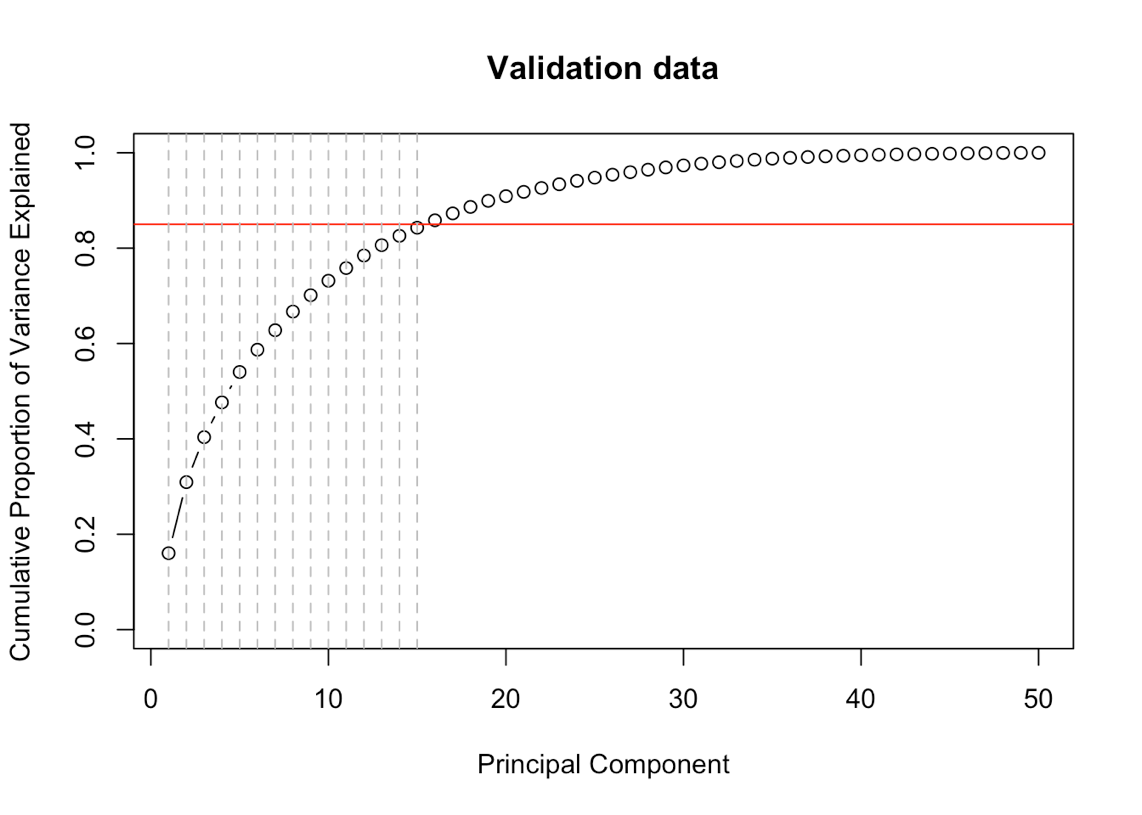
To deal with the missing values, we can choose those only the column with a 100% completion rate. Then, I export data2 as a tab delimited text file (newWLE.txt) that is ready for the following steps.

# discard the first 7 variables which appears to be not important   
data2 <- data1[colSums(is.na(data1))==0][-(1:7)]  
# remove rows with missing values  
sum(is.na(data2)) # no missing values now  
# export data2 as a tab delimited text file   
write.table(data2, "newWLE.txt", sep="\t")

### 4.2 Data Preparation for Model Selection

In this part, I will show the details on how train and validation data for model selections are generated.

library(caret) # need createDataPartition function from 'caret' package  
data <- read.csv("newWLE.txt", header=TRUE, sep='\t')  
# use the original five categories instead of two catergories in the midterm report  
# length(which(data$classe=="A")) # 11159  
# length(which(data$classe=="B")) # 7593  
# length(which(data$classe=="C")) # 6844  
# length(which(data$classe=="D")) # 6432  
# length(which(data$classe=="E")) # 7214  
# dim(data) # 39242 51  
# partition data into training and test data (7:3)  
set.seed(1200)  
inTrain <- createDataPartition(y=data$classe, p=.7,list=F)  
train <- data[inTrain,]; valid <- data[-inTrain,]  
  
Y.train <- as.matrix(train[,dim(train)[2]])   
Y.valid <- as.matrix(valid[,dim(valid)[2]])  
n.train <- dim(train)[1] # sample size 27472  
n.valid <- dim(valid)[1] # sample size 11770  
k.train <- length(unique(Y.train)) # number of classes is 5 (A B C D E)  
k.valid <- length(unique(Y.valid)) # number of classes is 5 (A B C D E)  
  
# scale the training data and then scale the validation data using the mean and variance of the training data.  
x.train.mean <- apply(train[,-51], 2, mean)  
x.train.sd <- apply(train[,-51], 2, sd)  
# standardize training data to have mean 0 and sd 1  
X.train <- t((t(train[,-51])-x.train.mean)/x.train.sd)   
# apply(X.train, 2, mean) # check zero mean and they are 0s  
# apply(X.train, 2, sd) # check sd and they are 1s  
  
# standardize validation data using training mean and sd  
X.valid <- t((t(valid[,-51])-x.train.mean)/x.train.sd)   
  
pca.train <- princomp(X.train, cor=T) # principal components analysis using correlation matrix  
# summary(pca.train) # the first 15 components consist of ~ 85% of varaiance  
pr.var <- pca.train$sdev^2; pve <- pr.var/sum(pr.var)  
plot(cumsum(pve), xlab="Principal Component", ylab="Cumulative Proportion of Variance Explained", main="Training data", ylim=c(0,1),type='b')  
abline(h=0.85,col="red"); abline(v=1:15,col="grey",lty=2)  


pc.comp.train <- pca.train$scores  
pc.comp1.train <- pc.comp.train[,1]; pc.comp2.train <- pc.comp.train[,2]   
pc.comp3.train <- pc.comp.train[,3]; pc.comp4.train <- pc.comp.train[,4]   
pc.comp5.train <- pc.comp.train[,5]; pc.comp6.train <- pc.comp.train[,6]   
pc.comp7.train <- pc.comp.train[,7]; pc.comp8.train <- pc.comp.train[,8]  
pc.comp9.train <- pc.comp.train[,9]; pc.comp10.train <- pc.comp.train[,10]   
pc.comp11.train <- pc.comp.train[,11]; pc.comp12.train <- pc.comp.train[,12]   
pc.comp13.train <- pc.comp.train[,13]; pc.comp14.train <- pc.comp.train[,14]   
pc.comp15.train <- pc.comp.train[,15]  
X.train <- cbind(pc.comp1.train, pc.comp2.train, pc.comp3.train, pc.comp4.train, pc.comp5.train, pc.comp6.train, pc.comp7.train, pc.comp8.train, pc.comp9.train, pc.comp10.train, pc.comp11.train, pc.comp12.train, pc.comp13.train, pc.comp14.train, pc.comp15.train)  
data.train <- as.data.frame(cbind(Y.train, X.train))  
names(data.train) <- c("Y", "X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8", "X9", "X10", "X11", "X12", "X13", "X14", "X15")  
pca.valid <- princomp(X.valid, cor=T) # principal components analysis using correlation matrix  
# summary(pca.valid) # the first 15 components consist of ~ 85% of varaiance  
pr.var1 <- pca.valid$sdev^2; pve1 <- pr.var1/sum(pr.var1)  
plot(cumsum(pve1), xlab="Principal Component", ylab="Cumulative Proportion of Variance Explained", main="Validation data", ylim=c(0,1),type='b')  
abline(h=0.85,col="red"); abline(v=1:15,col="grey",lty=2)  
  
pc.comp.valid <- pca.valid$scores  
pc.comp1.valid <- pc.comp.valid[,1]; pc.comp2.valid <- pc.comp.valid[,2]   
pc.comp3.valid <- pc.comp.valid[,3]; pc.comp4.valid <- pc.comp.valid[,4]   
pc.comp5.valid <- pc.comp.valid[,5]; pc.comp6.valid <- pc.comp.valid[,6]   
pc.comp7.valid <- pc.comp.valid[,7]; pc.comp8.valid <- pc.comp.valid[,8]   
pc.comp9.valid <- pc.comp.valid[,9]; pc.comp10.valid <- pc.comp.valid[,10]   
pc.comp11.valid <- pc.comp.valid[,11]; pc.comp12.valid <- pc.comp.valid[,12]   
pc.comp13.valid <- pc.comp.valid[,13]; pc.comp14.valid <- pc.comp.valid[,14]   
pc.comp15.valid <- pc.comp.valid[,15]  
X.valid <- cbind(pc.comp1.valid, pc.comp2.valid, pc.comp3.valid, pc.comp4.valid, pc.comp5.valid, pc.comp6.valid, pc.comp7.valid, pc.comp8.valid, pc.comp9.valid, pc.comp10.valid, pc.comp11.valid, pc.comp12.valid, pc.comp13.valid, pc.comp14.valid, pc.comp15.valid)  
data.valid <- as.data.frame(cbind(Y.valid, X.valid))  
names(data.valid) <- c("Y", "X1", "X2", "X3", "X4","X5", "X6", "X7", "X8", "X9", "X10", "X11", "X12", "X13", "X14", "X15")  
  
write.table(data.train, "train.txt", sep="\t")  
write.table(data.valid, "valid.txt", sep="\t")

### 5 Model Selection and Evaluation

### 5.1 Model a: Multinomial logit regression (MR) analysis using multinom function in the nnet package

library(caret) # need confusionMatrix function in 'caret' package

## Loading required package: lattice  
## Loading required package: ggplot2

data.train <- read.csv("train.txt", header=TRUE, sep='\t')  
data.valid <- read.csv("valid.txt", header=TRUE, sep='\t')  
library(nnet) # package that can do multinomial Log-Linear Models  
model.lr.train <- multinom(Y ~ ., data.train) # Fits multinomial log-linear models using multinom function

pred.class.train <- predict(model.lr.train)   
confusionMatrix(pred.class.train, data.train$Y) # confusion matrix for training data

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 5328 1401 2195 631 881  
## B 327 1989 417 714 816  
## C 659 581 1283 459 396  
## D 1025 740 545 2204 689  
## E 473 605 351 495 2268  
##   
## Overall Statistics  
##   
## Accuracy : 0.4758   
## 95% CI : (0.4699, 0.4818)  
## No Information Rate : 0.2844   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3292   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.6820 0.3742 0.2678 0.48945 0.44911  
## Specificity 0.7402 0.8974 0.9076 0.86943 0.91419  
## Pos Pred Value 0.5105 0.4666 0.3798 0.42360 0.54103  
## Neg Pred Value 0.8542 0.8567 0.8544 0.89676 0.88050  
## Prevalence 0.2844 0.1935 0.1744 0.16391 0.18382  
## Detection Rate 0.1939 0.0724 0.0467 0.08023 0.08256  
## Detection Prevalence 0.3799 0.1552 0.1230 0.18939 0.15259  
## Balanced Accuracy 0.7111 0.6358 0.5877 0.67944 0.68165

# The prediction accurancy for the five classes ranges from ~ 0.6 to ~ 0.7; sensitivity ranges from 0.3 to 0.7; specificity ranges from 0.7 to 0.9.  
  
model.lr.valid <- multinom(Y ~ ., data.valid) # Fits multinomial log-linear models using multinom function

pred.class.valid <- predict(model.lr.valid)   
confusionMatrix(pred.class.valid, data.valid$Y) # confusion matrix for validation data

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2361 614 868 241 310  
## B 146 831 129 306 346  
## C 261 268 660 175 190  
## D 452 331 276 1031 264  
## E 127 233 120 176 1054  
##   
## Overall Statistics  
##   
## Accuracy : 0.5044   
## 95% CI : (0.4953, 0.5135)  
## No Information Rate : 0.2844   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3666   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.7054 0.3650 0.32148 0.5345 0.48706  
## Specificity 0.7586 0.9023 0.90800 0.8656 0.93171  
## Pos Pred Value 0.5373 0.4727 0.42471 0.4380 0.61637  
## Neg Pred Value 0.8663 0.8556 0.86365 0.9046 0.88966  
## Prevalence 0.2844 0.1935 0.17443 0.1639 0.18386  
## Detection Rate 0.2006 0.0706 0.05607 0.0876 0.08955  
## Detection Prevalence 0.3733 0.1494 0.13203 0.2000 0.14528  
## Balanced Accuracy 0.7320 0.6337 0.61474 0.7000 0.70939

# The prediction accurancy for the five classes ranges from ~ 0.6 to ~ 0.7; sensitivity ranges from 0.3 to 0.7; specificity ranges from 0.7 to 0.9.

### 

### Model b: Linear discriminant analysis (LDA) using lda function in MASS package

library(MASS)  
model.lda.train <- lda(Y ~ ., data.train)  
pred.lda.train <- predict(model.lda.train)$class   
confusionMatrix(pred.lda.train, data.train$Y)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 5237 1354 2083 539 826  
## B 412 2081 502 915 868  
## C 720 641 1300 500 356  
## D 1037 695 560 2139 767  
## E 406 545 346 410 2233  
##   
## Overall Statistics  
##   
## Accuracy : 0.4728   
## 95% CI : (0.4669, 0.4788)  
## No Information Rate : 0.2844   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3265   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.6704 0.39146 0.27134 0.47502 0.44218  
## Specificity 0.7557 0.87827 0.90225 0.86682 0.92387  
## Pos Pred Value 0.5217 0.43554 0.36963 0.41150 0.56675  
## Neg Pred Value 0.8523 0.85745 0.85427 0.89387 0.88029  
## Prevalence 0.2844 0.19351 0.17440 0.16391 0.18382  
## Detection Rate 0.1906 0.07575 0.04732 0.07786 0.08128  
## Detection Prevalence 0.3654 0.17392 0.12802 0.18921 0.14342  
## Balanced Accuracy 0.7131 0.63487 0.58680 0.67092 0.68302

# The prediction accurancy for the five classes ranges from ~ 0.6 to ~ 0.7; sensitivity ranges from 0.3 to 0.7; specificity ranges from 0.7 to 0.9.

model.lda.valid <- lda(Y ~ ., data.valid)  
pred.lda.valid <- predict(model.lda.valid)$class   
confusionMatrix(pred.lda.valid, data.valid$Y) # confusion matrix for validation data

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2290 581 837 223 316  
## B 204 880 157 330 357  
## C 318 288 664 216 158  
## D 446 302 273 1000 272  
## E 89 226 122 160 1061  
##   
## Overall Statistics  
##   
## Accuracy : 0.5008   
## 95% CI : (0.4918, 0.5099)  
## No Information Rate : 0.2844   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3629   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.6842 0.38647 0.32343 0.51840 0.49030  
## Specificity 0.7677 0.88960 0.89915 0.86861 0.93785  
## Pos Pred Value 0.5392 0.45643 0.40389 0.43611 0.63993  
## Neg Pred Value 0.8595 0.85806 0.86283 0.90197 0.89092  
## Prevalence 0.2844 0.19346 0.17443 0.16389 0.18386  
## Detection Rate 0.1946 0.07477 0.05641 0.08496 0.09014  
## Detection Prevalence 0.3608 0.16381 0.13968 0.19482 0.14087  
## Balanced Accuracy 0.7259 0.63804 0.61129 0.69351 0.71407

# The prediction accurancy for the five classes ranges from ~ 0.6 to ~ 0.7; sensitivity ranges from 0.3 to 0.7; specificity ranges from 0.8 to 0.9.

### 

### Model c: Quadratic discriminant analysis (QDA) using qda function in the MASS package

model.qda.train <- qda(Y ~ ., data.train)  
pred.qda.train <- predict(model.qda.train)$class   
confusionMatrix(pred.qda.train, data.train$Y) # confusion matrix for training data

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 3700 449 147 100 61  
## B 208 2332 257 46 226  
## C 2819 1290 3970 1413 555  
## D 876 274 263 2609 638  
## E 209 971 154 335 3570  
##   
## Overall Statistics  
##   
## Accuracy : 0.589   
## 95% CI : (0.5832, 0.5948)  
## No Information Rate : 0.2844   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4898   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.4736 0.43868 0.8286 0.57939 0.7069  
## Specificity 0.9615 0.96674 0.7321 0.91071 0.9256  
## Pos Pred Value 0.8302 0.75986 0.3951 0.55987 0.6814  
## Neg Pred Value 0.8213 0.87772 0.9529 0.91697 0.9334  
## Prevalence 0.2844 0.19351 0.1744 0.16391 0.1838  
## Detection Rate 0.1347 0.08489 0.1445 0.09497 0.1300  
## Detection Prevalence 0.1622 0.11171 0.3657 0.16963 0.1907  
## Balanced Accuracy 0.7176 0.70271 0.7804 0.74505 0.8162

# The prediction accurancy for the five classes ranges from ~ 0.7 to ~ 0.8; sensitivity ranges from 0.4 to 0.8; specificity ranges from 0.7 to 0.9.  
  
model.qda.valid <- lda(Y ~ ., data.valid)  
pred.qda.valid <- predict(model.qda.valid)$class   
confusionMatrix(pred.qda.valid, data.valid$Y) # confusion matrix for validation data

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2290 581 837 223 316  
## B 204 880 157 330 357  
## C 318 288 664 216 158  
## D 446 302 273 1000 272  
## E 89 226 122 160 1061  
##   
## Overall Statistics  
##   
## Accuracy : 0.5008   
## 95% CI : (0.4918, 0.5099)  
## No Information Rate : 0.2844   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3629   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.6842 0.38647 0.32343 0.51840 0.49030  
## Specificity 0.7677 0.88960 0.89915 0.86861 0.93785  
## Pos Pred Value 0.5392 0.45643 0.40389 0.43611 0.63993  
## Neg Pred Value 0.8595 0.85806 0.86283 0.90197 0.89092  
## Prevalence 0.2844 0.19346 0.17443 0.16389 0.18386  
## Detection Rate 0.1946 0.07477 0.05641 0.08496 0.09014  
## Detection Prevalence 0.3608 0.16381 0.13968 0.19482 0.14087  
## Balanced Accuracy 0.7259 0.63804 0.61129 0.69351 0.71407

# The prediction accurancy for the five classes ranges from ~ 0.6 to ~ 0.7; sensitivity ranges from 0.3 to 0.7; specificity ranges from 0.8 to 0.9.

### Model d: Random forest (RF) using randomForest function

Random Forest (RF) is one of the tree-based methods for regression or classification which involve stratifying or segmenting the predictor space into a number of simple regions. The basic idea of these tree-based methods is to partition the space and identify some representative centroids. The random forests algorithm is shown below. Let N be the number of observations and assume that the response variable is binary:

* Take a random sample of size N with replacement from the data (bootstrap sample).
* Take a random sample without replacement of the predictors.
* Construct a split by using predictors selected in Step 2.
* Repeat Steps 2 and 3 for each subsequent split until the tree is as large as desired. Do not prune. Each tree is produced from a random sample of cases, and at each split a random sample of predictors.
* Drop the out-of-bag data down the tree. Store the class assigned to each observation along with each observation's predictor values.
* Repeat Steps 1-5 a large number of times (e.g., 500).
* For each observation in the dataset, count the number of trees that it is classified in one category over the number of trees.
* Assign each observation to a final category by a majority vote over the set of trees. Thus, if 51% of the time over a large number of trees a given observation is classified as a "1", that becomes its classification.

RF method involves of randomly choosing subset of predictor variables as well as randomly choosing subset of data samples; it can dramatically reduce both model variance and bias.

library(randomForest); set.seed(1)

## randomForest 4.6-10  
## Type rfNews() to see new features/changes/bug fixes.

model.rf.train <- randomForest(Y ~ ., data.train, importance=TRUE)  
pred.rf.train <- predict(model.rf.train)   
confusionMatrix(pred.rf.train, data.train$Y) # confusion matrix for training data

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 7764 55 9 3 1  
## B 21 5208 34 0 11  
## C 22 42 4717 117 6  
## D 4 5 24 4377 9  
## E 1 6 7 6 5023  
##   
## Overall Statistics  
##   
## Accuracy : 0.9861   
## 95% CI : (0.9846, 0.9874)  
## No Information Rate : 0.2844   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9824   
## Mcnemar's Test P-Value : 5.036e-15   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9939 0.9797 0.9846 0.9720 0.9947  
## Specificity 0.9965 0.9970 0.9918 0.9982 0.9991  
## Pos Pred Value 0.9913 0.9875 0.9619 0.9905 0.9960  
## Neg Pred Value 0.9976 0.9951 0.9967 0.9945 0.9988  
## Prevalence 0.2844 0.1935 0.1744 0.1639 0.1838  
## Detection Rate 0.2826 0.1896 0.1717 0.1593 0.1828  
## Detection Prevalence 0.2851 0.1920 0.1785 0.1609 0.1836  
## Balanced Accuracy 0.9952 0.9884 0.9882 0.9851 0.9969

# The prediction accurancy for the five classes is ~ 0.99; sensitivity and specificity all are around 0.99.

model.rf.valid <- randomForest(Y ~ ., data.valid, importance=TRUE)  
pred.rf.valid <- predict(model.rf.valid)   
confusionMatrix(pred.rf.valid, data.valid$Y) # confusion matrix for validation data

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 3254 53 26 10 5  
## B 34 2173 25 5 16  
## C 30 38 1964 95 21  
## D 23 5 35 1815 14  
## E 6 8 3 4 2108  
##   
## Overall Statistics  
##   
## Accuracy : 0.9613   
## 95% CI : (0.9576, 0.9647)  
## No Information Rate : 0.2844   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.951   
## Mcnemar's Test P-Value : 1.692e-09   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9722 0.9543 0.9566 0.9409 0.9741  
## Specificity 0.9888 0.9916 0.9811 0.9922 0.9978  
## Pos Pred Value 0.9719 0.9645 0.9143 0.9593 0.9901  
## Neg Pred Value 0.9890 0.9891 0.9908 0.9885 0.9942  
## Prevalence 0.2844 0.1935 0.1744 0.1639 0.1839  
## Detection Rate 0.2765 0.1846 0.1669 0.1542 0.1791  
## Detection Prevalence 0.2845 0.1914 0.1825 0.1607 0.1809  
## Balanced Accuracy 0.9805 0.9729 0.9689 0.9665 0.9860

# The prediction accurancy for the five classes ranges from ~ 0.97 to ~ 0.99; sensitivity ranges from 0.94 to 0.97; specificity ranges from 0.98 to 0.99.

### 

### Conclusion

#### Results summary for the validation dataset for model selection  
# Method: Accuracy Sensitivity Specificity  
# MR: 61~73% 32~71% 76~93%  
# LDA: 61~73% 32~68% 87~94%  
# QDA: 61~73% 39~68% 77~94%  
# RF: 97~99% 94~97% 98~99% (BEST)

From the above summary table, we can clearly see that the **random forest** method yields significantly higher prediction accuracy, sensitivity and specificity, thus, I would prefer to using this method to predict future potential manners with new testing data. The other three methods actually are not that bad as their accuracies are higher than 60% and specificities are higher than 75%; however, their sensitivities are low for some classes (~30%).