C8Project

Yuchao

November 4, 2016

Project Introduction

In this project, we use the data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways, and these ways corresponded to five classes as A, B, C, D, E. The goal of this project is to **predict** the manner in which users did the exercise.

Data Pre-processing

Before any data analytics and model fitting, let's load the dataset and go over the available attributes first. Both the training and testing datasets are directly read from online source.

```
library(data.table)
training <- fread("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
testing <- fread("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
training <- as.data.frame(training)
testing <- as.data.frame(testing)
nattr <- ncol(training)
lattr <- last(names(training))
```

There are totally 160 columns in the data set, while the last one is *classe*, the target we're going to predict. Based on the introduction, we will use the data from accelerometer attached on the body, so we only need a subset of attributes that is relevant to accelerometer readouts and the target class.

```
ACC <- grepl('acc', names(training))
nacc <- sum(ACC)
ACC[2] <- TRUE
ACC[length(ACC)] <- TRUE
acctrain <- training[,ACC]
acctest <- testing[,ACC]
head(acctrain)
```

```
##   user_name total_accel_belt var_total_accel_belt accel_belt_x
## 1  carlitos               3                   NA          -21
## 2  carlitos               3                   NA          -22
## 3  carlitos               3                   NA          -20
## 4  carlitos               3                   NA          -22
## 5  carlitos               3                   NA          -21
## 6  carlitos               3                   NA          -21
##   accel_belt_y accel_belt_z total_accel_arm var_accel_arm accel_arm_x
## 1            4           22              34            NA        -288
## 2            4           22              34            NA        -290
## 3            5           23              34            NA        -289
## 4            3           21              34            NA        -289
## 5            2           24              34            NA        -289
```

```
## 6            4           21              34            NA        -289
##   accel_arm_y accel_arm_z total_accel_dumbbell var_accel_dumbbell
## 1         109        -123                   37                NA
## 2         110        -125                   37                NA
## 3         110        -126                   37                NA
## 4         111        -123                   37                NA
## 5         111        -123                   37                NA
## 6         111        -122                   37                NA
##   accel_dumbbell_x accel_dumbbell_y accel_dumbbell_z total_accel_forearm
## 1             -234               47             -271                  36
## 2             -233               47             -269                  36
## 3             -232               46             -270                  36
## 4             -232               48             -269                  36
## 5             -233               48             -270                  36
## 6             -234               48             -269                  36
##   var_accel_forearm accel_forearm_x accel_forearm_y accel_forearm_z classe
## 1                NA             192             203            -215      A
## 2                NA             192             203            -216      A
## 3                NA             196             204            -213      A
## 4                NA             189             206            -214      A
## 5                NA             189             206            -214      A
## 6                NA             193             203            -215      A
```

```
naTrain <- round(sum(is.na(acctrain$var_total_accel_belt))/length(acctrain$var_total_accel_belt),4) * 1
naTest <- round(sum(is.na(acctest$var_total_accel_belt))/length(acctest$var_total_accel_belt),4) * 100
acctrain <- acctrain[,!grepl('var',names(acctrain))]
acctest <- acctest[,!grepl('var',names(acctest))]
nattr <- ncol(acctrain) - 2
```

It is observed that among the 20 attributes relevant to accelerometer, all the attributes named following "var" contain NA value. In fact, 97.93% of these attributes are NA in the training set, while 100% are NA in the testing set. Therefore, we exclude the attributes named following "var" from both the training and testing datasets. Finally, we have our dataset with 16 attributes of accelerometer, one attribute for the subject and one for the target classes.

To be able to evaluate the accuracy of different models generated from the labeled dataset, we will separate the training set into two datasets without overlap, one is used for model training while the other is used for validation. The model with the best performance will be used to predict the exercise manner in the testing dataset in the end.

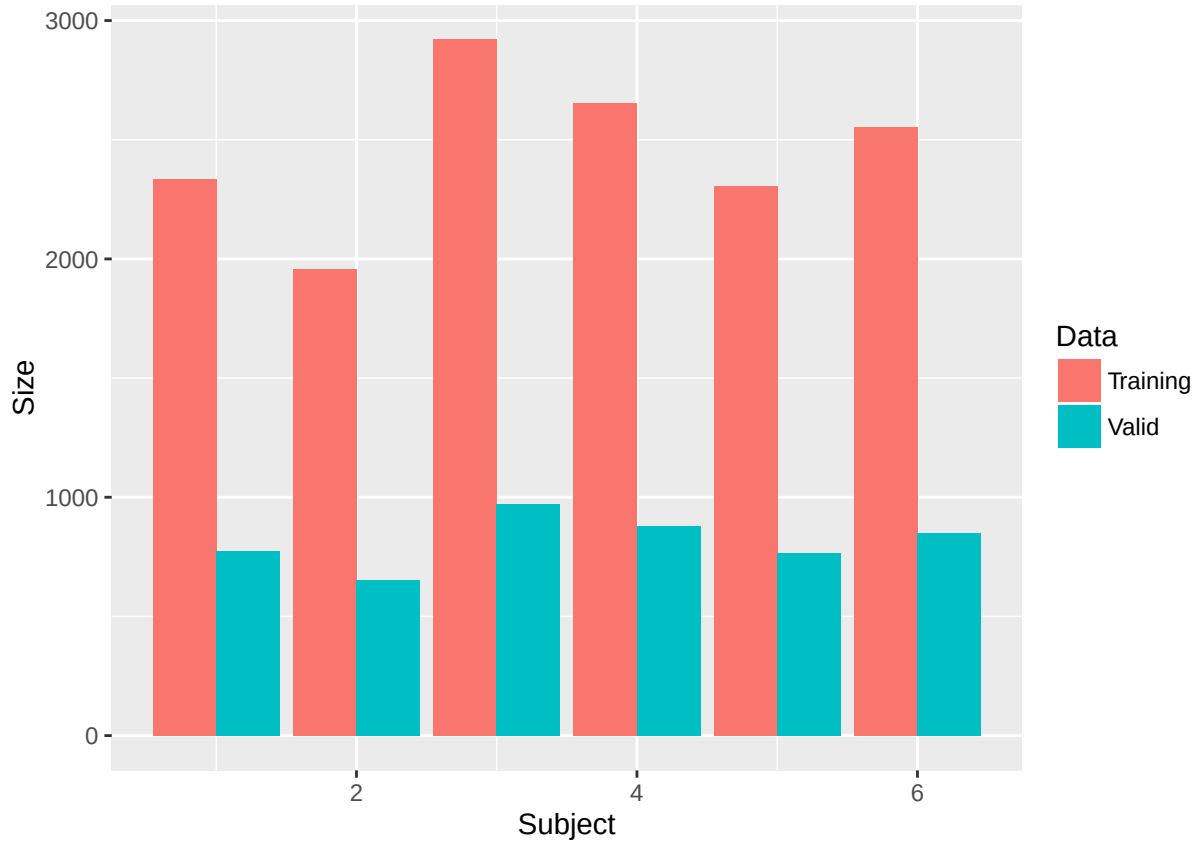
```
library(ggplot2)
library(caret)
```

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

```
nlist = c()
unilist = unique(acctrain$user_name)

training <- data.frame(matrix(nrow = 1, ncol = NCOL(acctrain)))
names(training) <- names(acctrain)
valid <- data.frame(matrix(nrow = 1, ncol = NCOL(acctrain)))
names(valid) <- names(acctrain)
```

```
for (i in c(1:length(unilist))){
  S <- acctrain[acctrain$user_name == unilist[i],]
  nlist[i] = nrow(S)
  intrain <- createDataPartition(y = S$classe, p = 0.75, list = FALSE)
  sTr <- S[intrain,]
  sTe <- S[-intrain,]
  training <- rbind(training, sTr)
  valid <- rbind(valid, sTe)
}

training <- training[-1,]
valid <- valid[-1,]

trSize = c()
vaSize = c()

for (i in c(1:length(unilist))){
  S <- training[training$user_name == unilist[i],]
  trSize[i] = nrow(S)
  S <- valid[valid$user_name == unilist[i],]
  vaSize[i] = nrow(S)
}

ptSize <- data.frame(Data = c(rep('Training',length(unilist)),rep('Valid',length(unilist))), Subject = 
```

Because the training dataset involves the data from 6 subjects, to separate it into two sets, we'd like to have a balanced partition for all the subjects. Therefore, we first create data partition with 75% training and 25% valid for each subject's subset, and then combine the data from all the subjects. The figure below shows the size of the training and valid dataset corresponding to each subject. It is worth mentioning that these two datasets are all from the original training set, and the testing set is kept untouched.



Prediction by Multi-Subject Learning

With these two datasets, we first build prediction model based on **multi-subject** learning, by which means we will predict the exercise manner of each subject using the model trained by the data from not only that specific subject, but also the other subjects.

Because the size of this training dataset is not trivial, Random Forest and GBM take too long time to response. We choose several light-weighted algorithm for classification, namely Decision Tree, Linear Discriminant Analysis and K Nearest Neighbor. We will validate the performance on each subject's data separately.

```
ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3);
set.seed(8844)

modDT <- train(classe~., data = training[,-1], method = "rpart", trControl = ctrl);
```

```
## Loading required package: rpart
```

```
modLDA <- train(classe~., data = training[,-1], method = "lda", trControl = ctrl);
```

```
## Loading required package: MASS
```

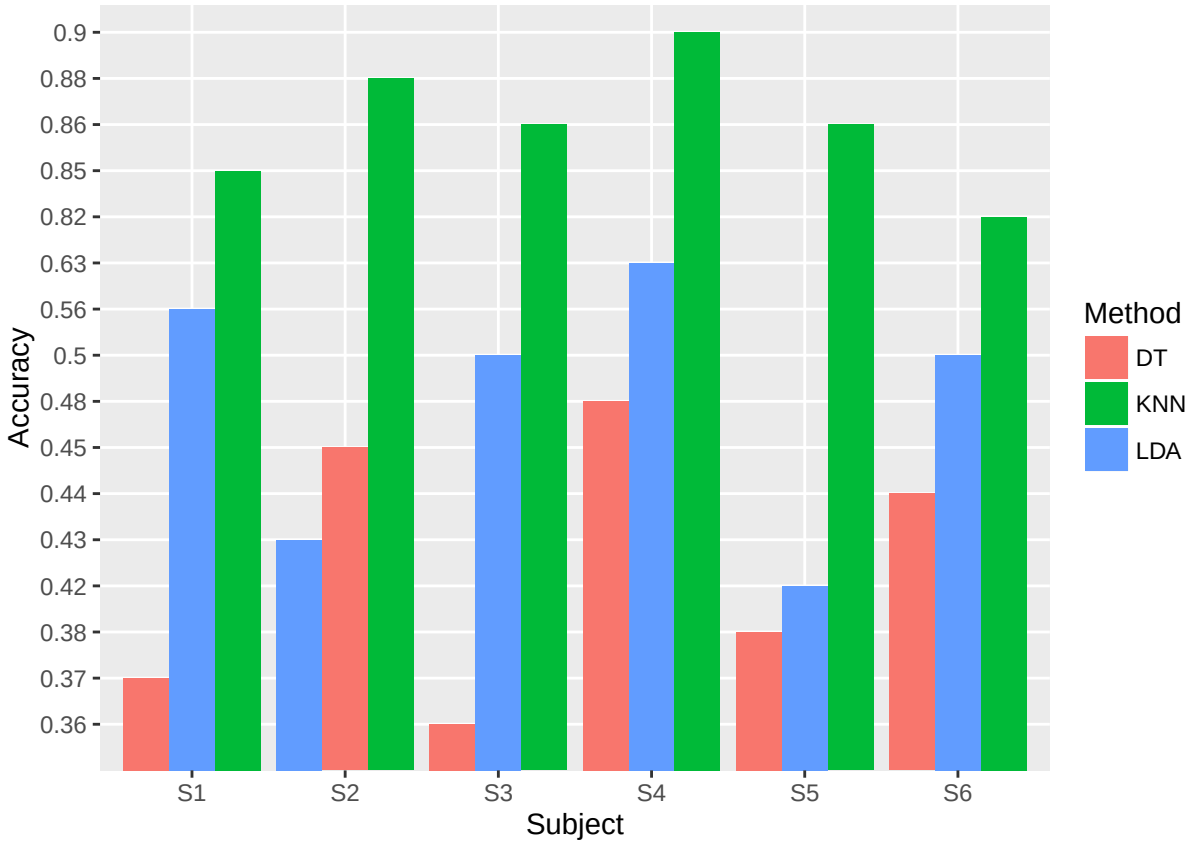
```
modKNN <- train(classe~., data = training[,-1], method = "knn", trControl = ctrl);
```

```
acc1 <- matrix(nrow = 3, ncol = length(unilist))
colName <- c()
for(s in c(1:length(unilist))){
  colName <- c(colName,sprintf('S%d',s))
  sub_valid <- valid[valid$user_name == unilist[s],]

  p1 <- predict(modDT, newdata = sub_valid[,-1])
  C <- confusionMatrix(sub_valid$classe, p1)
  acc1[1,s] <- C$overall[1]

  p2 <- predict(modLDA, newdata = sub_valid[,-1])
  C <- confusionMatrix(sub_valid$classe, p2)
  acc1[2,s] <- C$overall[1]

  p3 <- predict(modKNN, newdata = sub_valid[,-1])
  C <- confusionMatrix(sub_valid$classe, p3)
  acc1[3,s] <- C$overall[1]
}

colnames(acc1) <- colName
row.names(acc1) <- c("DT","LDA","KNN")
acc1
```

```
##            S1        S2        S3        S4        S5        S6
## DT  0.3698454 0.4546851 0.3614830 0.4790011 0.3838120 0.4440518
## LDA 0.5605670 0.4347158 0.4963955 0.6299659 0.4242820 0.5041225
## KNN 0.8479381 0.8786482 0.8568486 0.9012486 0.8642298 0.8209658
```

Here gives the accuracy using different methods tested on each individual's validation dataset. The figure below shows the same result. Among the three methods, KNN achieves a better result comparing with the others, which is reasonable considering different people have different exercise patterns and KNN will find the same subject's instance with small distance.

```
acc_m <- matrix(nrow = 18, ncol = 3)
acc_m[,1] <- rep(colName,3)
acc_m[,2] <- c(rep("DT",6),rep("LDA",6),rep("KNN",6))
acc_m[,3] <- c(acc1[1,],acc1[2,],acc1[3,])
acc_m[,3] <- round(as.numeric(acc_m[,3]),2)
colnames(acc_m) <- c("Subject","Method","Accuracy")
ggplot(data = as.data.frame(acc_m), aes(x = Subject, y = Accuracy, fill = Method))+geom_bar(stat="ident
```



Prediction by Single-Subject Learning

In this section, we train the model based on each subject separately, and apply the model to only the corresponding subject's validation dataset. Ideally, the model trained by the same person's data should work better than the model trained with multiple persons.

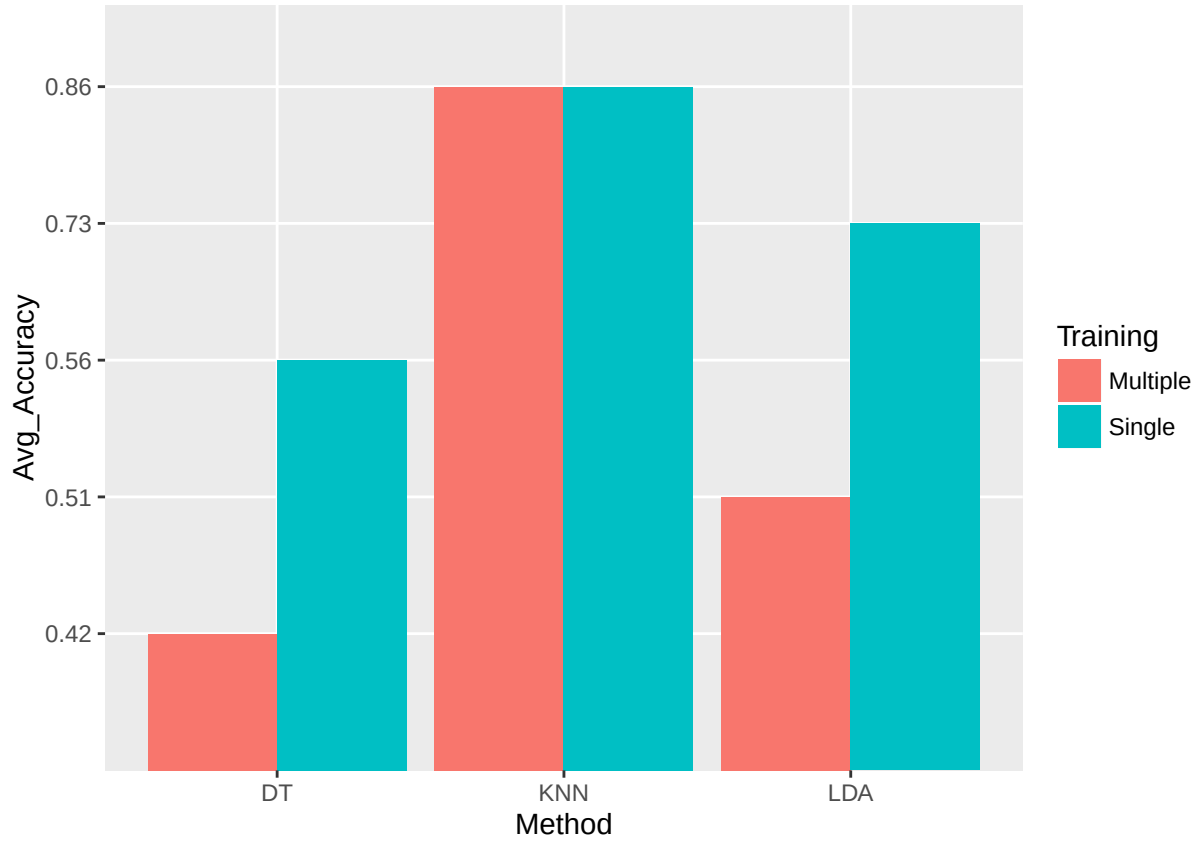
```
acc2 <- matrix(nrow = 3, ncol = length(unilist))

for(s in c(1:length(unilist))){
  sub_train <- training[training$user_name == unilist[s],]
  modDT <- train(classe~., data = sub_train[,-1], method = "rpart", trControl = ctrl);
  modLDA <- train(classe~., data =sub_train[,-1], method = "lda", trControl = ctrl);
  modKNN <- train(classe~., data = sub_train[,-1], method = "knn", trControl = ctrl);

  sub_valid <- valid[valid$user_name == unilist[s],]

  p1 <- predict(modDT, newdata = sub_valid[,-1])
  C <- confusionMatrix(sub_valid$classe, p1)
  acc2[1,s] <- C$overall[1]

  p2 <- predict(modLDA, newdata = sub_valid[,-1])
  C <- confusionMatrix(sub_valid$classe, p2)
  acc2[2,s] <- C$overall[1]

  p3 <- predict(modKNN, newdata = sub_valid[,-1])
```

```
  C <- confusionMatrix(sub_valid$classe, p3)
  acc2[3,s] <- C$overall[1]
}
colnames(acc2) <- colName
row.names(acc2) <- c("DT","LDA","KNN")
acc2
```

```
##            S1        S2        S3        S4        S5        S6
## DT  0.4858247 0.4700461 0.4799176 0.7877412 0.6005222 0.5488810
## LDA 0.6958763 0.7450077 0.6230690 0.8558456 0.7624021 0.7173145
## KNN 0.8453608 0.8740399 0.8599382 0.9046538 0.8655352 0.8268551
```

The results show the increase in accuracy of all the three models by training with the specific subject's data. The figure below shows the average accuracy over all the six subjects by two types of training approach. Because the performance of KNN is the best overall, and the accuracy based on two training approaches is very close, we will predict the exercise manner in the testing set using KNN model trained by all the training data.

```
acc_comp <- matrix(nrow = 6, ncol = 3)
acc_comp[,1] <- rep(c("DT","LDA","KNN"),2)
acc_comp[,2] <- c(rep("Multiple",3),rep("Single",3))
acc_comp[1:3,3] <- rowMeans(acc1)
acc_comp[4:6,3] <- rowMeans(acc2)
acc_comp[,3] <- round(as.numeric(acc_comp[,3]),2)
acc_comp[,3] <- round(as.numeric(acc_comp[,3]),2)
acc_comp <- as.data.frame(acc_comp)
names(acc_comp) <- c("Method","Training","Avg_Accuracy")
ggplot(data = acc_comp, aes(x = Method, y = Avg_Accuracy, fill = Training))+geom_bar(stat="identity", p
```



The prediction results of these 20 samples are shown as below.

```
modKNN <- train(classe~., data = training[,-1], method = "knn", trControl = ctrl);
predResult <- predict(modKNN, newdata = testing[,-1])
predResult
```

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